The Motor Industry Software Reliability Association

MISRA c/o Electrical Group, MIRA, Watling Street, Nuneaton, Warwickshire, CV10 0TU, UK.
Telephone: (024) 7635 5290. Fax: (024) 7635 5070. E-mail: misra@mira.co.uk Internet: http://www.misra.org.uk

Report 1

Diagnostics and Integrated Vehicle Systems

February 1995

PDF version 1.0, January 2001

Acknowledgements

The following MISRA members contributed to this report :

Klaus Allen	Rover Group
Ian Kendall	Jaguar Cars
Keith Longmore	Lotus Engineering
Tony Moon	AB Automotive
Mike Radford	Lucas Electronics
Chris Shakespeare	Rover Group
Lloyd Thomas	AB Automotive

We are also grateful for the support of and contributions from:

Neil Andrewartha	}
Alex Abbot	} Rolls Royce and Associates
Vivien Hamilton	}
Ken Tindell	York University

Contents

1. Integrated Vehicle Systems

1.1 Scope

This report covers the aspects of vehicle engineering which relate to the use of software to support integrated communications and diagnostics networks. The subjects addressed in this report are whole vehicle architecture, communications and multiplexing, on-board diagnostics, off-board diagnostics, tools and testing. The aim is to review the wider aspects of automotive engineering which can directly influence the software which is eventually embedded in vehicles.

1.2 Background

In today's environment, where modern vehicles are increasing their reputation for performance, quality and value for money, succeeding in the market place is ever more dependent on the ability to employ new technology in all aspects of the business. Over the last decade the electrical feature content in vehicles has approximately doubled every 3 to 4 years, and this trend is still continuing. Increasing competition in the crowded market place creates strong competitive pressures for ever more features, quality and reliability. However this has to be achieved for less cost, weight and in a much reduced time scale.

Motor vehicle manufacturers originate from a traditional mechanical engineering biased industry, but are increasingly having to cope with the explosion in specialist electronic content. This is true at all levels from design, through manufacture, to service and it affects both the vehicle manufacturers and their suppliers. As an example, even the conventional wiring harnesses on a high specification vehicle are very large, requiring some 2000 plus wires.

As more and more features are added, the complexity of subsystems has multiplied and the interaction and interfaces between subsystems has reached the limit of traditional methods. Of particular relevance to MISRA is the fact that software has now become a necessary element in the deployment of new vehicle features. A further recent development is the introduction of in-vehicle serial data buses and multiplex systems for control and diagnostics functions.

Complexity is a two edged sword which makes the ability to find real faults unaided increasingly difficult. Modern systems tend to have fall-back modes which in some cases can be outwardly indistinguishable to normal operation. This would occur where safety is not compromised, however there may still be environmental effects. The potential for damage to the environment would result in the requirement that the fault be rectified as soon as reasonably possible - this is the rationale behind OBD II legislation for exhaust emissions in the USA.

The industry is being pulled in two directions. Firstly the need for multiplexed / integrated electrical systems to assist with wiring and interactions, and secondly the need for vastly improved diagnostics capability to assist in fixing problems.

1.3 Recommendations

1.3.1 Whole Vehicle Architecture

- The objectives should be clearly laid down before the start of a project. For example the nature of a project will be different for a volume production system, when compared to a research and development prototype.

- The project definition should consist of a list of vehicle features and functions to be implemented. It should be agreed and documented by the appropriate authorities of the vehicle manufacturer and made available to the design and development teams before detailed work begins.

- The project definition should also include any known legislative requirements and project assumptions (eg. the use of an existing engine and/or ECU in a new vehicle).

- It is particularly important to lay down and agree which features are fixed and which are customer options.

- Strict change control should apply from project definition onwards.

- Any changes made to the project definition will lead to increased risk from both safety and commercial points of view. The later changes are made the greater the incremental risk.

- Before attempting to write requirements specifications for individual ECUs, particularly for integrated systems, an architecture of the complete vehicle electrical system should be defined in conceptual form. The objective is to ensure that the requirements specifications for sub-systems and components are consistent and compatible with one another. In defining an architecture, teams of component engineers, system engineers and management should work together to give a broad view of the whole vehicle electrical system.

- Be very clear about the objectives for a vehicle design before entering an architecture exercise and ensure that this is agreed with the area in the company responsible for product definition and strategy. Agree the documented deliverables for the architecture study.

- Results from the safety assessment should be fed into the design at an early stage. The architectural design is at least as important as the detailed design stages in

minimising the effort required to meet safety requirements. The development of an optimal architecture will involve several iterations**.**

- Where appropriate electrical isolation should be considered for functions needing a high level of integrity. This will mean that effort can be focused more towards demonstrating compliance with safety requirements.

- To ensure all aspects are covered, split the work into functional and implementation aspects, sometimes called logical and physical partitioning, and address each with a team. Close cooperation and communication between the architecture teams is essential.

- For logical partitioning, create and maintain a model of the functional requirements of the complete vehicle electrical system. Split functional requirements modelling into 4 phases :
    - analysis of high level requirements
    - definition of system boundary and environment
    - detailed top-down modelling of functional requirements
    - the implementation of functions on vehicle hardware.

- For physical partitioning, consider implementation (sensors and actuators, etc.) at an early stage to allow the architecture to encompass other vehicle design aspects. (eg. packaging and body design). Identify the interfaces required by each subsystem (ie. i/o identification). Agree the available technology options which may affect the architecture, for example communications protocols and switching (relays/FETs). Review these options as the architecture study progresses.

- Keep up to date lists of all current assumptions and open or unresolved issues made during the architectural design stage. Keep these under version control.

- If multiple suppliers are to be used for sourcing programmable components which must be integrated, then it becomes essential that the vehicle manufacturer performs a proper functional analysis and maintains an overall system specification. This ensures that the component specifications given to each supplier are compatible.

- Strict change control procedures should be enforced for software, hardware and other appropriate design material at all times. Changes should be subject to review by the architecture teams responsible.

- Use the appropriate tools to aid in the capture of requirements, in particular consider CASE tools to support functional modelling during the definition of a vehicle architecture.

- The design of an integrated system, with a range of integrity levels, should aim to segregate functions of different integrity levels. Otherwise, the highest level of integrity should prevail for the whole system.

1.3.2 Communications and Multiplexing

- If the analysis of requirements highlights the potential for using a bussed architecture, where possible an internationally recognised protocol standard should be used for network communications, eg. ISO/DIS 11898 (CAN High Speed), ISO/DIS 11519 (VAN and CAN Low Speed), SAE J1850.

- Proprietary protocols are designed with a specific application in mind, where some special attribute such as cost, speed or integrity dominates. Vehicle manufacturers should assure themselves that proprietary protocols are adequate for their intended use.

- As an alternative to dedicated hardware for lower speed communications systems, it is possible to support some protocols in software with a suitable microcontroller.

- Where communication is necessary for systems requiring a high level of integrity, it is recommended that a dedicated bus is used. The use of a high integrity protocol should be considered, e.g. CAN, VAN, J1850, etc.

- Because of the differing requirements of some sub-systems, for example those for real-time control and driver/passenger services, multiple networks can be used. If a gateway is required, it should be designed in as part of each system. The data transfer requirements for the gateway can be determined during the final stage of functional requirements modelling, and should aim to minimise the transfer traffic.

- System responsibility should be assigned to a defined person or body, usually the vehicle manufacturer, who is responsible for the definition and integration of the communications network. The person or body should also maintain close control over, and ensure good communications between, the different design teams that use the multiplex bus. Component suppliers should work closely together and accept constraints on their software design.

- Communications and multiplex systems should be specified and designed using a layered model of the network services, for example ISO-OSI 7 layers. For most automotive systems a three layer model consisting of physical, transfer and application layers is adequate.

- The physical layer specifications should ensure that interfaces between ECUs are defined and adhered to. Where possible use an existing or a common standard.

- The transfer layer should carry out message transmission scheduling as well as checks on received message timeouts and corruption (eg. CRC).

- The application layer should be designed to perform range and/or rate-of-change checks on data associated with high integrity functions (plausibility checks). These are normally carried out at the point of reception, but can also be applied before transmission.

- Scheduling of message transmissions is important to ensure that the correct data rates or cycle times can be achieved. It recommended that the transmission software drivers are not fragmented within the application code. It is better to have a centralised scheduling algorithm, based on one timing source in each ECU, which can be analyzed for predictability.

- Ensure that the relationships between the communications layers in each ECU are defined and adhered to.

- It is recommended that loading on the bus, message priorities, delay (latency) times and target processes are carefully analyzed to ensure that the system operates reliably. The network should be robust against situations such as overloading and excessive delay, and avoid deadlocking.

- To achieve the higher integrity levels, it may be necessary to augment the basic protocol with additional features such as:
    - performing repeated transmissions of data to enable voting before an action is taken;
    - providing extra redundancy in the data field to offer error detection and/or correction codes
    - time tagging each message.

1.3.3 On-Board Diagnostics

- On-board Diagnostics should seek to:
    - ensure that the vehicle is maintained in as safe and as legal a state as possible;
    - advise the driver of significant failures;
    - provide information to authorised personnel for prompt fault identification;
    - keep the customer mobile for as long as possible.

- Diagnostic software should not be added as an afterthought: it should be included as part of the overall system requirements and design strategy. It should be clear what the objective of the diagnostic scheme is.

- In principle, on-board diagnostic software can be used to detect three kinds of fault:
    - non-existent or incorrect sensor signals;
    - actuators not performing as intended;

- processes within the host system not functioning as specified, where the malfunctions are not due to the processor itself.

· If a detection strategy is to be effective it should:
  - detect only "genuine" faults;
  - invoke limp-home states in a safe manner;
  - provide a warning to the driver in a reasonable fashion;
  - store the fault information and make it accessible to repair personnel.

  For these requirements to be met, it is essential to clearly lay down and understand the criteria for each part of the detection and diagnosis process.

· In systems where performance degrades as components age and wear, on-board condition monitoring can be employed to raise system integrity by identifying potential failures before they occur. The extent of monitoring will depend on the severity of the potential hazard, ie the higher the integrity requirements the more comprehensive the condition monitoring that needs to be applied.

· Detected faults should result in one or more of a range of actions:
  - warn the driver;
  - store fault data till reset; store fault data semi-permanently;
  - set limp-home mode semi-permanently (until repaired);
  - set local default states.

· Depending on the seriousness of a fault, a limp-home state may be invoked to allow the vehicle to continue its journey in a safe manner but with some form of performance or functional restriction. It is advisable to maintain as much functionality as possible and not unnecessarily degrade driveability.

· Limp-home should only be invoked when there is no other alternative, and if it is going to significantly impact on the driver’s control of the vehicle, adequate notice of the event should be given, as far as this is possible.

· Consideration should be given to the mechanism of warnings, so that the driver is not unduly alarmed, or unnecessarily warned for minor faults. Such over-warning can lead to confusion or a nonchalant attitude towards heeding the warning.

· Failure management should offer alternative sensor information or mechanical back up wherever possible, and all default definitions should be supported by a well reasoned diagnostic strategy and objectives.

· Safe states should be derived with reference to the system hazard analysis. The default action taken must be appropriate for the controllability category of the hazards involved.

- Hazard analysis of default states should consider potential driving situations, and how the default states, or combinations of default states, interact with those situations. Hazard analysis should also consider the effects of system reset, so as to maintain a safe state.

- It is recommended that on-board diagnostics and tools are available at the first release of software to assist the development process and calibration, and not just viewed as post production support.

- The software for supporting off-board diagnostic tools which communicate with on-board systems should perform the following:
  - transmit sensor data on demand;
  - transmit fault codes on demand;
  - allow adjustment of authorised on-board parameters by the diagnostic tool;
  - provide a sufficient level protection against unauthorised access

1.3.4 Tools and Testing

- The networking of CASE tools can be a significant advantage for project teams, and should be considered to aid the configuration management of design material.

- A CASE tool should be capable of good graphical support for the preferred methodology and be well supported by a good data dictionary.

- Carefully assess CASE tool suppliers for stability, long term commitment and support services.

- Assess new tools to ensure that they support and integrate into the existing development process rather than hinder it. Cumbersome tools and processes tend to be circumvented.

- CASE tools cost more than the purchase price alone. Allow for maintenance, training and integration of the tool into the development process.

- Do not take the decision to switch design methodologies lightly. If possible assess alternative methodologies in parallel with existing methods on a current project.

- Before beginning design the rules and naming conventions for the use of the CASE tool must be defined, agreed and documented.

- The true power of a toolset can only be realised when these tools are integrated into a common environment.

- It is currently recognised that a high degree of assurance in a design may be obtained by using formal mathematical methods and by providing all supporting proofs which have been checked with a proof checker.

- Use simulation tools to predict communications network performance during the analysis and design phases. Compare actual performance against the prediction so that future simulations can be made more accurate.

- To meet integrity requirements defensive techniques should be used where appropriate.

- All system resources used by a high integrity level function should be provided up to the same standard as that function. Note that redundancy and diversity may be applicable to reduce the integrity requirements of individual parts.

- The choice of programming language itself should be strongly influenced by the way that language is used and by the availability of tools which exist to support it.

- Plan integration testing activities in advance. Do not test in an ad-hoc fashion; ensure each test has a purpose defined in the plan. It must remain the responsibility of the vehicle manufacturer to plan, perform and approve system integration tests.

- The integration test plan should aim to ensure that the system is built such that functions at lower levels are first tested individually.

- Assess individual ECUs for compliance with their interface specifications before attempting to perform integration tests.

- Test the entire software aims to ensure that all modules interact correctly. Coverage analysis should be used to ensure that the testing adequately exercises the code structurally.

- Each module should be tested separately and then, after integration, tests should be made on the entire system.

1.3.5 Off-Board Diagnostics

- Off-board diagnostic systems should have the ability to:
    - retrieve data from on-board systems;
    - apply on-board tests;
    - diagnose relevant faults that cannot be detected by on-board systems.

- If a test routine invoked by a service bay tool can affect the normal operation of the system, then care should be taken in ensuring the unit is configured back to normal operation before ending the diagnostic session. Exiting the diagnostic session, either by controlled exit or power down, must return the unit to normal operation.

- Service diagnostics should be designed to assist both expert and inexperienced technicians. For inexperienced technicians it may be appropriate to lead them down a fault tree until the failed component is identified (lowest replaceable unit).

- If using the communications network as a medium for performing diagnosis, it is also necessary to consider failure of the communications network itself.

- Access to appropriate information will be required for non-franchised repairers and for roadside repair throughout the life of the vehicle.

- Technical information proprietary to the manufacturer requires support in terms of training and backup services.

- Elaborate off-board diagnostic systems are not a substitute for engineering reliability and robustness into the vehicle itself.

- Diagnostic tools should be able to identify the lowest replaceable unit (LRU). The tool can also be used to remind the technician not to attempt to repair an LRU.

- Information on warranty claims should be available at all levels in both the vehicle manufacturers' and the component suppliers' organisations, providing a feedback mechanism for design decisions.

- For bussed diagnostic systems, a common protocol and common message definition should apply to all systems connected to the bus. Where possible use an international standard, for example ISO TC22 WG1, "Key Word Protocol 2000" (KWP 2000).

- Consideration should be given to the use of event recorders that can be fitted to vehicles to store dynamic parameters to assist in locating intermittent faults.

- Legislation is in force in the USA (OBD II), and is likely to be proposed in Europe in the future, to enforce diagnostic requirements related to exhaust emissions. Thoroughly understand the provisions of OBD II for the US and KWP 2000 for Europe.

- Software must be taken as a serious contributor to quality and reliability performance by the vehicle manufacturer and not dismissed as a supplier's problem.

2. Whole Vehicle Architecture

2.1 Introduction

Traditional electrical engineering in an automotive company was, and still is in some areas, based on component engineering, where one engineer would take total responsibility for one component. This worked well when the electrical system consisted only of a starter, alternator, headlights, wipers and a radio, but the complexity of modern vehicles demands a different approach.

In order to handle the increase in electrical features, both from a physical packaging point of view and cost, automotive engineers are using multiplex technology and software based control. Electronic modules are becoming general purpose processing elements communicating on a data bus, with the features and functions distributed around an integrated system.

The so-called electrical/electronic architecture of a vehicle must therefore encompass many elements of the component engineering approach, but pull them all together into a complete system which has a clear structure. In order to define a proper architecture for a complex vehicle, teams of component engineers must work together with systems engineers and management to give a complete perspective on the electrical system, and this must happen before the requirements for the software and hardware are considered for individual Electronic Control Units, or ECUs. The opportunity for a clean sheet design is rare, and often many of the components and other aspects of a vehicle electrical system are predetermined due to factors outside the control of the designers, e.g. cost, company policy, mechanical constraints, etc. The process for defining an architecture must accommodate influences such as these, even if this deviates from the ideal. Engineering a vehicle is a complex operation involving many people, each of whom often have different priorities.

2.2 Vehicle Architectural Partitioning

If a vehicle is to benefit from having an integrated and optimised electrical system the process of engineering it must be approached in a way which permits all the various elements to come together correctly. Vehicle architecture like any large problem should be broken down or partitioned into smaller problems. Architectural partitioning should be both functional and physical and aim to ensure that the requirements that are defined for each sub-system’s hardware and software fit into an overall vehicle structure completely and consistently. An architectural partitioning exercise should precede any attempt to jump into individual sub-system in detail. The result should be a coherent and consistent set of requirements specifications.

Ideally, the initial partitioning process should only consider abstractions of the features and functions required of the vehicle, without attempting to make decisions about implementation in hardware or software. However, in a real vehicle design situation it is necessary to define many of the hardware aspects early on to enable packaging engineers to reserve the required

space envelopes and body engineers to proceed with the design of the metal panels. In addition it is usual for large proportions of the system to be already known, for example when making use of existing engines or gearboxes. Hence although considering implementation details in the early stages of architectural partitioning contradicts the ideal approach, it is essential to ensure that the process is relevant to the overall vehicle programme.

Physical partitioning is concerned with the feasibility and optimisation of hardware and will generally result in a series of both dedicated and multi-functional ECUs which can provide all the i/o required by the system. The strategies for interconnection, fusing, power switching and the installation in the vehicle are all aspects of physical partitioning.

Logical partitioning is concerned with the functionality of the system and hence is mainly concerned with software which must execute on the hardware (although a clear hardware/software split may not be appropriate in the early stages). Before a system can be logically partitioned (ie. functions distributed between ECUs), the functional requirements for the complete vehicle system must be analyzed and modelled in detail.

The physical and logical partitioning can be handled by two separate teams of engineers, but the work is highly interactive and decisions are often inter-dependent, hence good inter-team communications and cooperation are essential. A close working relationship between the teams may be assisted by having some engineers who participate in both activities.

2.2.1 Physical Partitioning -Implementation

The objective of physical partitioning is to define the hardware required to meet the needs of the system and hence the features to be offered by the electrical system must be known and agreed before beginning the design. The result is a set of hardware requirements specifications for the ECUs which form the vehicle architecture.

The basic approach is to analyze iteratively each subsystem in turn to make up the whole vehicle system, e.g. Engine Management, Transmission Control, Lighting, Memory Seats/Mirrors, Security, Locking, Wipers, etc. The team must have sufficient knowledge to be aware of the possible interactions between subsystems, and therefore should have a good spread of specialist knowledge of the different subsystems. Additional expertise can be brought in when required.

Often there are a number of technology options which can affect the way in which it is possible to define an architecture (eg. packaging constraints, harness routing, design standards, etc.), and it may be necessary to consider these before an implementation configuration can be looked at in detail. Examples of such options include whether "SmartFET" are to be considered as an alternative to relays, or whether full multiplexing ("smart" sensors / actuators / switches - full distributed control) can supplement serial switching (communicating modules - some localised control).

The most likely approach is to run a series of regular team meetings. Each team session can be structured in a similar way, starting with lists of known and given properties of the

subsystem, then two or three alternative layouts can be considered (for that subsystem only) which can be subjectively rated by the team according to a number of criteria such as cost, complexity, ease of assembly, weight, reliability, etc.

The starting point should be the signals and information which are required to perform the sub-system functions and where each might come from, i.e. i/o requirements. (This is equivalent to the system context but extends to possible physical location of the source/destination of each signal/piece of data, e.g. wheel speed sensors must be located near each wheel). The ECUs should not be discussed until after the i/o is understood. Each session should finish with a preferred architecture for that subsystem together with a list of assumptions and open issues associated with it. (There are always plenty of unanswered questions at this stage).

When all the subsystems have been dealt with at least once, then it is likely to be necessary to revisit some or all of them again. As these iterations are carried out, the optimised implementation of the complete integrated system should begin to develop. It is difficult to be prescriptive about this process as it depends heavily on company working practices and the nature of the project.

A final few sessions are often required to complete this process and make sure all the assumptions are valid and all the open issues are closed.

The result of the implementation configuration study should be a feasible topology for the vehicle electrical system. Topology is the term given to the way the system is laid out and it can most simply be expressed by a plan view of the vehicle showing the number and locations of the ECUs, sensors and actuators, the routing and connectivity of the communications busses and main wiring harnesses, and the location of related items such as fuse boxes and relays. The detailed design of individual electronic modules can proceed from the topology as the architecture of the hardware interfaces are now known.

2.2.2 Logical Partitioning - Functional Requirements Modelling

This is concerned with the creation and maintenance of a model of the functional requirements of the complete vehicle electrical system, and the result is a set of requirements specifications for each of the individual ECUs which make up the system.

The process can be based around a suitable real-time structured analysis technique such as the Ward-Mellor [1] or the Hately-Pirbhai [2] methodology and can be implemented on a CASE tool. Whichever methodology is used should be a top-down approach and the support of a computer tool is highly recommended to help manage the large amounts of information which are generated as the detail emerges.

The CASE tool should be capable of good graphical support for the preferred methodology, and be well supported by a data dictionary which can be used to analyze and structure the information at all stages of the modelling process. A good database, or data dictionary, is important to allow logical sorting and access to the elements of the model. Before beginning

analysis/modelling the rules and conventions for the use of the data dictionary of the CASE tool must be defined, agreed and documented.

Functional requirements modelling can be split into 4 phases. Some of the terminology used here is from the Ward-Mellor methodology, but this does not affect the validity of the activities described below if other methodologies are used.

a) Analysis of the high-level system requirements is important to tie down exactly what the customer will want the system to do, and the starting point for this is an outline features list, or functional specification for the vehicle which has been agreed by the product definition department of the vehicle manufacturer. One very important aspect in capturing requirements is to lay down and agree whether features are fixed or customer options, as this can significantly affect the architectural outcome.

b) Following a thorough high-level system requirements analysis an "environmental" model can be started. This defines the boundary of the whole vehicle system, and from this it is possible to begin to generate the interfaces (ie. the i/o given the known actuators and sensors), to components which are regarded as external to the functional requirements. The environmental model therefore shows the environment or context in which the system exists, and is sometimes called the context diagram.

c) The next step is to consider the "behavioural" model. This takes the top level environmental model from b) and refines it through a number of lower levels using data-flow diagrams, state-transition diagrams, logic tables, etc. to satisfy each of the requirements defined in a). The use of a CASE tool for this enables consistency to be checked at each level and links the diagrams with the data dictionary. This step contains most of the detailed functional analysis and definition and hence is the most time consuming.

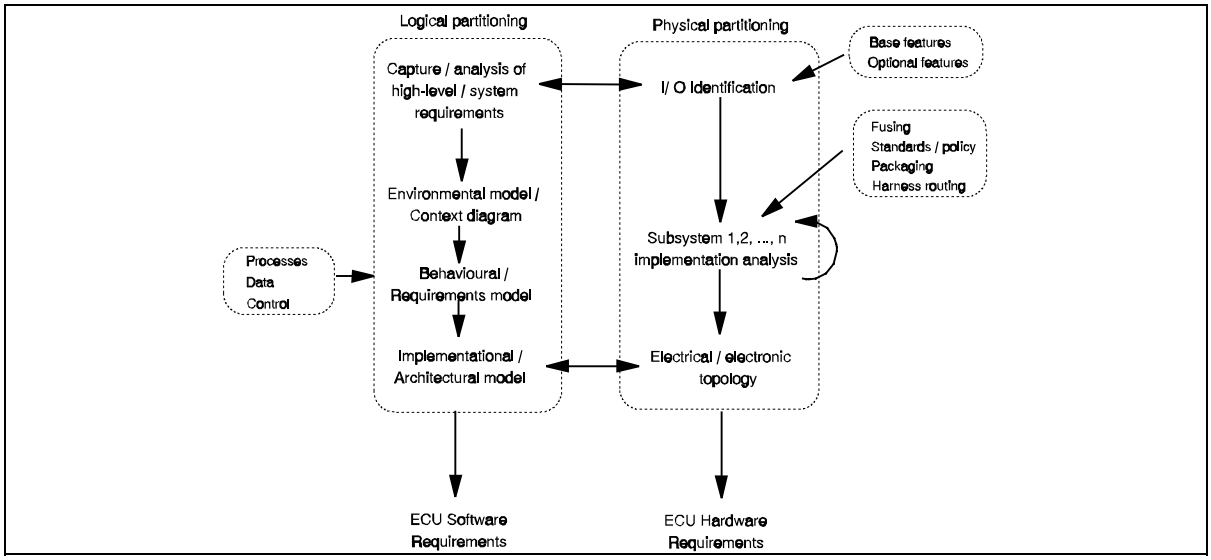


Figure 1 - Logical and Physical Partitioning

d) Once the behaviour of the system has been defined in detail, the final step is to consider real implementation by feeding in the results of physical partitioning, see Figure 1. This basically involves mapping the behavioural model onto the topology to give a so called "implementation" model. As this must involve the decision of which ECUs perform which functions, then the communication and interconnection requirements must also be defined as at this stage. It may also be beneficial to use the information on size and complexity now available to construct a cost model for the complete system

Due to the amount of work involved, a team approach will almost certainly have to be adopted for functional requirements modelling. Such a team can be very efficient once the first couple of levels of the model are completed, as the work is naturally partitioned and each team member can address a different aspect in parallel. This is why it is important to use naming conventions, etc, and one team member must be responsible for the integration of the work of the others into a master model. The local area networking of CASE tools would prove a significant advantage in this respect.

It is worth noting that a significant proportion of the output of a functional requirements model, as well as playing a vital role in the definition of an architecture, will also serve to generate a good proportion of the documentation necessary that may be required by a possible future full vehicle type approval process. Hence the work overhead in meeting such regulations, should they become mandatory, are reduced.

2.3 Multiple Networks and Gateways

The electrical system architecture will depend heavily on the market sector that the vehicle is intended for, and on the number of features and customer options. The current situation is that in basic vehicles, where hardware is limited by cost and functions are simple, a

multiplexing network may not be required at all, and if it is, it may only be a single, relatively low speed communication bus (10-100 Kbits/sec) linking some or all of the ECUs. However in high specification or complex vehicles with many features (including trucks), the optimum architecture may require one or more high speed multiplex busses (100-1000 Kbits/sec) [3].

A further partition can arise when the information exchange requirements and the role of the connected subsystems are considered. It is usual for there to be heavy communication traffic between major powertrain and dynamic control related subsystems such as engine control, transmission control, suspension control, ABS, etc. and this communication mainly consists of time-critical data of a physical nature (load, speed, position, etc.). Conversely, there is also a significant amount of traffic between subsystems which deal with occupant comfort and convenience features, such as seat movement, mirrors, lights, etc. Here the communication is much less time critical and is mainly switching/binary in nature (ie. things being turned on and off).

Hence there is a natural partition between the Powertrain/Vehicle Dynamics Systems and the Body Systems, each with different communications requirements - high speed, time critical "physical" data for the former, and lower speed, less time critical "switching" data for the latter. If the communications are optimised for the different applications it is likely to lead to different operating speeds, bus loads and protocols, which may be incompatible depending on optimisation parameters, company design policy and device availability.

It is evident that this partition will not give two mutually exclusive subsystems, but that there will be a need to transfer some data from one network to the other. This gives rise to the concept of a gateway. The gateway should be capable of communicating on both the powertrain/vehicle dynamics (high speed) and body system (low speed) networks, and as such must be designed in as part of both systems. The data transfer requirements for the gateway are determined during the final stage of functional requirements modelling, and the actual gateway ECU can be selected so as to minimise the data transfer traffic through it. The gateway selection decision will require close cooperation between the implementation and functional aspects of the system.

As well as the powertrain and body systems networks mentioned above, there is another case where multiple network solutions may be used in an architecture. This is the situation where a dedicated communications link between two ECUs is used to perform a specific function for a specific reason. The four most common reasons for considering a dedicated link as part of a vehicle architecture are: to reduce the bus loading where the function exceeds the capacity of a general purpose link; where the physical layer is not compatible with a common bus standard; to meet the integrity requirements of a function where it might be unwise to use a general purpose bus; or finally, where the cost is lower.

2.4 Specifications for Integrated Systems

The results of the implementation study (physical partition) and functional model (logical partition) will form the core of the hardware and software requirements specifications for the individual ECUs in the system. The iterative approach to hardware integration and the top-down approach to functional analysis can help to minimise problems of consistency and compatibility in individual component specifications, because they should all be derived from the same sources of information. It is therefore possible to have high confidence in the specifications for the ECUs and their interfaces used in the system, and in the interconnection between them.

One of the potential offshoots of this approach is that the information forming an architecture can be used directly for the formation of off-board diagnostic strategies. Traditionally such information had to be collected later on in the design lifecycle and was always at risk of being out of date, wrong or incomplete. The use of architecture results for this purpose could lead to big savings in resources and overall development costs.

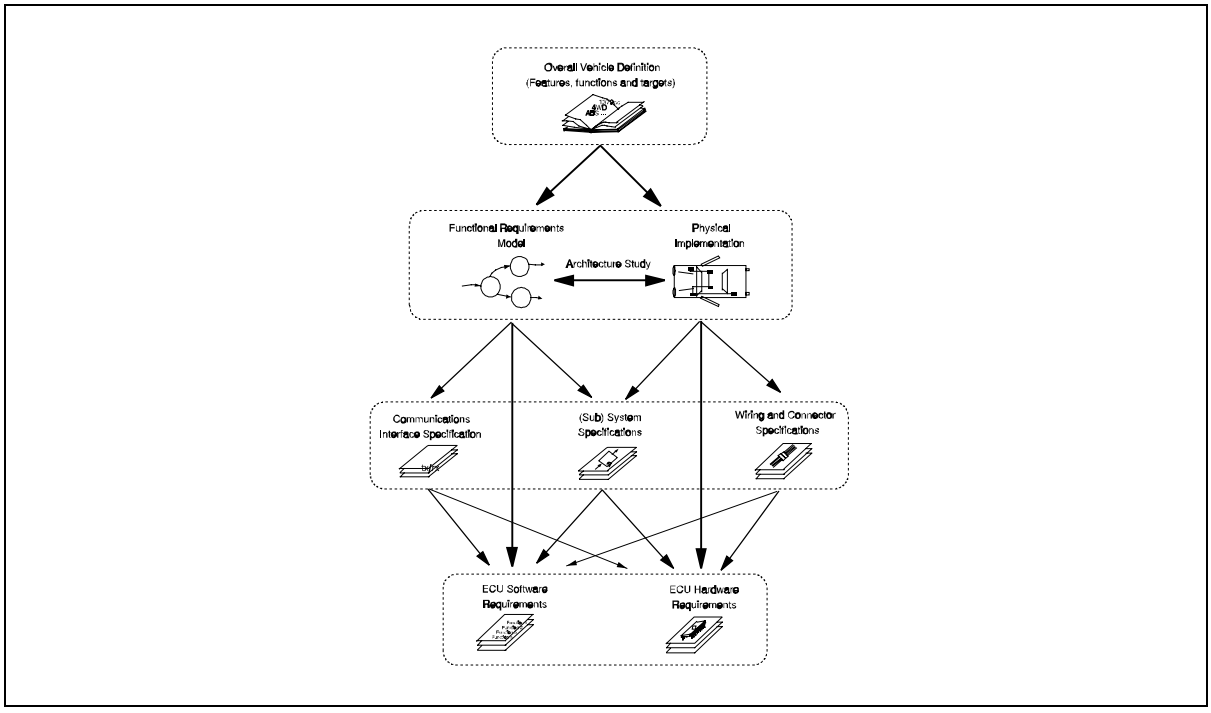


Figure 2 - Example Specification Hierarchy

It is strongly recommended that the partitioning approach adopted during architecture conceptual design is also employed during the specification stage, that is a structured approach to specifications is considered. For example it is better to have those aspects which relate to the system (such as the communications requirements) in a separate specification rather than repeat those requirements in each of the component specifications. This also reduces the chance of inconsistencies occurring. An example of a typical hierarchy of specifications is shown in Figure 2.

The partitioning of functions between the various electronic control units should have been decided in the last phase of functional requirements modelling and it should be stressed that this partitioning is critical for the successful design of multiplexed systems. The individual ECU software requirements specifications can be fed into the top of the software development lifecycle and be treated as separate items, each one being subject to further requirements analysis and design before code is written. If multiple suppliers are to be used for sourcing the various programmable components, then it becomes even more important that the vehicle manufacturer performs a proper functional analysis and maintains an overall system specification such that the subsystem specifications given to each supplier are consistent with one another.

Strict change control procedures must be enforced for both the software and the hardware at all times with any changes being subject to review by the architecture teams responsible. The use of CASE tools for the functional modelling has enormous benefits in assisting change control, as all parts of the system which may be affected by a change can be ascertained quickly and easily.

2.5 Security and Tampering

A multiplexed architecture of a motor vehicle offers considerable advantages in terms of flexibility. However this gives rise to an increased risk of tampering by unauthorised persons. Steps must be taken to ensure that it is difficult to tamper with the vehicle electrical system, and to improve the detectability of tampering should it occur. (See MISRA Report 4-Software in Control Systems). This can be done in a number of ways such as :

- making the bus physically difficult to access (eg. use fibre optics);
- restricting the use of the communications bus to non-safety critical features;
- using encrypted security messages on the bus to identify non-original components and software.

The flexibility of the bus can also be used to enhance the anti-theft security offered by the vehicle. For example, such enhancements can take the form of sophisticated and distributed digital engine immobilisation algorithms, or the use of Vehicle Identification Codes being received, transmitted and compared by every electronic module connected to the bus (see section 6.7.2, Vehicle Characterisation).

2.6 Integrity Levels and Architecture

Safety engineering activities [4-5] can be carried out as part of an architecture process without causing large amounts of additional work. Hazard analysis can be performed as part of the discussion on each subsystem in the implementation study, and when the topology is defined

more detailed Failure Mode and Effects Analysis and Fault Tree Analysis can be performed as the architecture refines and develops. This will enable redundancy to be added into the most critical parts of the system to achieve failure rate targets at an early stage in the design. The approach for the hardware is not trivial but it is well understood and justifiable by statistics based on failure probability distributions (eg. Weibull, Exponential).

Because software does not exhibit failure by the same mechanism as hardware, the usual approach is to develop the software with a process according to an integrity level that gives sufficient confidence in the function it is required to perform. (see MISRA Report 2 - Integrity). However in a multiplexed system the integrity requirements for software need more careful consideration because of its distributed nature. Each function can be categorised for safety integrity easily enough, but a function may be performed by two or more programmable components in different locations within the vehicle. The most obvious method is to develop the whole system's software to the highest integrity level, but this is patently an approach which will lead to the problem of the majority of the system being over-engineered with all the associated costs of doing that.

2.6.1 Integrity Isolation

A hazard analysis will be performed to assign an integrity level to each of the functions. This will be based upon the effect of failure on controllability (see MISRA Report 2 - Integrity). During the architectural design phase, the functions will be allocated to ECUs. The architectural design phase will also determine where communication links will be, and how many there will be. Note that a single function may be split over more than one ECU.

After an initial vehicle architecture is defined, it will be necessary again to consider the hazards. The design may well have introduced new failure modes. For example, where a function is split between more than one ECU, the failure of a communication link between them may result in the function being in an indeterminate state. Where functions share resources, i.e. they use the same communications or reside on the same processor, they may interfere with one another.

To overcome these new failure modes, and to provide defence against any other hazards which the initial architecture did not address, it will probably be necessary to modify the design and possibly to include additional design elements.

Particular care will be necessary for the most critical functions, i.e. the highest integrity levels. In some cases, it might be decided that physical isolation and redundancy are the only acceptable design solutions for functions which require this level of integrity.

One method that might be applied at this stage is the PASSPORT Cross (see MISRA Report 2 - Integrity). This should provide a simple method of identifying areas of interaction. Designing to minimise the problem may prove to be cheaper overall, rather than accepting additional failure modes, which must be considered for all future modifications.

Defensive features, although effective, add complexity and additional cost to the design. A design which removes the problem should therefore be considered favourably compared with one that requires additional defensive features. Also, verification and validation issues are perhaps more important for an architecture designed to accommodate different integrity levels.

Physical isolation is to be preferred from the integrity point of view, provided that the required reliability and functionality can be achieved.

2.6.2 Verification and Validation Issues

The functions which have most effect on controllability will be required to have a high degree of verification and validation applied to them. Additional verification and validation procedures are required where such functions share resources with functions of lower integrity.

Sharing occurs where functions use the same communications, memory, data, functionality, or reside on the same processor. Where such sharing occurs it is necessary either to develop and verify all functions to the highest level of integrity required or to demonstrate non-interference. The degree of demonstration necessary will depend on:

- The level of integrity required of the highest function.
- The difference between the procedures required for the highest functions and those applied to the lower integrity functions.
- The degree of interference which might occur.

Potentially, the worst case occurs where a high integrity function resides on the same processor as a large and complex low criticality system, with which it shares data. Full development and verification of the low integrity system to the highest standards would be very expensive. An example may occur in the future for a road train system which perhaps shares a processor with part of the vehicle security system. Ideally, the architecture should be designed to avoid such a situation, but if this cannot be done for whatever reason, the following approach is suggested.

All resources used by the high integrity system should be considered to belong to the high integrity system. They should therefore be developed and verified to the same standards as the high integrity system. This includes shared functionality, input-output, data, memory and timing. In practise, this will require that the scheduler is developed as part of the high integrity system. The scheduler must have the capability to delay scheduling of the low integrity system, if to do otherwise would interfere with the high integrity system.

The possibility that a fault in the low integrity system could lead to a failure in the high integrity system must be considered. Typically, these faults arise from use of a weakly typed language (such as Assembler or C), or the misuse of the language (such as an array index out of range, failure to terminate a loop, etc.).

The feasibility and cost of performing the verification will be significantly affected by the coding and design standards which have been applied. Partitioning techniques which reduce interaction between different parts of the software i.e. modular designs, packages in Ada, etc. should be enforced.

It is possible to design systems using interrupts which can be verified to a high degree of confidence. However, it should be noted that to date this has been done only for small systems, and only where the entire system is subject to very detailed and extensive verification. Where interrupts must be used, careful scrutiny of all parts of the design is required, to ensure that interruption can not leave any part of the system in an indeterminate state (see MISRA Report 3 - Noise, EMC and Real-time). In practise, this is likely to lead to a requirement for very extensive static and dynamic modelling of all parts of the system.

Static analysis is capable of detecting control flow anomalies, data flow anomalies and discrepancies in data transformation, which could lead to faults (see MISRA Report 6 - Verification and Validation). It is possible to apply such static analysis without attempting to prove the software against its specification, which would be more expensive. The use of strongly typed languages, such as Pascal and Ada, and the avoidance of pointers and direct memory addressing make this task easier.

The software can also be verified dynamically, by using code assertions during the testing. Selecting sufficient and appropriate assertions is however a non-trivial task, and might best be accomplished using static analysis. Additionally, for the assertions to be useful, extensive testing will be needed, which is likely to be excessive compared to the requirements for that integrity level.

System testing should conform to the procedures recommended for the high integrity system. System tests need not consider combinations of inputs for the low integrity system, but should take place in a realistic environment. That is tests for the overall functionality of the high integrity system should be performed with the other systems performing realistic workloads. Stress tests for memory, timing and input-output should also be considered.

Defensive measures, as described below can also be included. In general, these should be treated as part of the high integrity system and developed and verified to the same standards.

2.6.3 Defensive Measures

Design features included as defensive measures will normally need to be classified as the same integrity as the highest function. For example, where a scheduler is included on a multi-function ECU, one possible failure mode is that it prevents one of the functions from operating, or disrupts its operation. This is equivalent to a total failure of that function, and hence the scheduler should be classed as requiring equivalent integrity as the highest function.

In some cases, the defensive feature only acts as a back-up, and hence it may be possible to classify it at a lower integrity. For example, the failure of a watchdog device to detect the failure of the function being monitored requires the failure of two independent components,

before it results in a fault. The probability of this double failure will be less than that of the failure of the monitored function alone, even if the watchdog is developed to a lower integrity.

Defensive features may be provided by additional software, hardware or mechanical features. They provide defence by introducing redundancy, diversity or detection capabilities.

Redundancy provides protection against physical damage, degradation or random failure, but does not protect against design faults. Redundancy requires the duplication of all or part of the system. It is applicable to mechanical or electrical or electronic systems. Duplication of an entire ECU will provide defence against damage to a unit or random hardware failures, but multiple copies of software will not protect against software errors.

Diversity provides defence against common mode failures. For example, an optical communications link is subject to different failure modes from an electronic data bus, which could be affected by EMC problems. Note however that diverse software implementations do not provide good protection against software design errors.

Detection at least ensures that the system does not continue to control from an undefined state or using erroneous data. Where detection is used, the behaviour of the system once a fault has been detected must be defined. This requires that at least a safe state is defined. Where possible recovery action, which restores the system to a known state and enables control to be resumed should also be defined. The case where the recovery action also fails should be considered. Detection systems include anti-bugging code and high level supervisors. These detect deviations from normal state and initiate actions to return to a safe state.

Fault tolerance can be achieved by use of fault detecting systems coupled with redundant or diverse systems. Dual or triple lane control systems are frequently used in other industries to achieve high integrity levels. The methods used to decide on overall control (e.g. voting or monitoring) need careful treatment, to ensure that they are not susceptible to a single failure. Fault tolerance is desirable in order to achieve better reliability. A system which detects a fault and stops may be safe, but is unlikely to be satisfactory to the end-customer, especially where a large network is involved. It should be possible for faults to be isolated in a small part, while the rest of the network continues to operate.

Systems suitable for on processor fault tolerance/defence include:

- Antibugging code-check inputs and outputs for each module.
- Anticorruption code-checksums, etc. This is especially useful where timeslicing is used.
- Code to check sequence of operations-detects unexpected code sequences, caused by design errors or corruption of the stack pointer.

- Order of magnitude checks-checks that values calculated are reasonable, for example of checking range and rate of change. This will detect serious corruption of data values.

The issue of security must also be considered. Deliberate tampering to change the performance of the vehicle may result in links between systems originally designed to be independent. Therefore, defences against tampering need to be considered during the hazards analysis process.

2.7 "Part Bins" and Re-use

In any vehicle programme there are always pressures to utilise existing parts taken from the "parts bin"; this may bring significant advantages in terms of cost and reduction of project risk. Although ideally this should not be allowed to compromise the design, in practice corporate strategy may dictate terms. The following points are useful guidelines which may help in rationalising this position.

- It is advantageous to separate "parts" in terms of their hardware and software content.

- Hardware should be commonised wherever possible, and it may be advantageous to carry redundancy to achieve this (depending on sales volume vs cost). For example, a piece of hardware may already exist and be suitable for a given application, except that it might have too many outputs. These outputs could be disabled and not used (Care needed - see MISRA Report 3-Noise, EMC and Real-Time). This may be cheaper than redesigning it with the correct number of outputs.

- Software is easy to modify in theory, but difficult to implement in practice, and is often underestimated. If parts containing bespoke software are to be reused in new configurations, and if modifications to software are necessary to achieve the required functionality, then this should not affect the lifecycle approach to the development of that software.

The modifications should be controlled from analysis of requirements, through design, coding, integration, validation and acceptance. Hazard analysis should be performed, and integrity requirements met including full verification throughout the process.

Alternatively a design philosophy using verified libraries is possible. This enables individual modules to be interchanged more easily. Proper integration testing is still required, but each one has been previously verified in isolation.

2.8 The Architecture Study Process

Problems in vehicle electrical system design are diverse, and therefore the studies into potential architectures will differ accordingly. The difference usually arises from the emphasis of the objectives for the design. For example, the objective may be to maximise reuse of existing components, or to minimise the cost, or to assess the impact of a new engine/vehicle configuration. For this reason it is important to be very clear about the objective of a design action before entering an architecture exercise, and ensure that this is agreed with the area in the company responsible for product definition and strategy.

Likewise the deliverables for the architecture study should be agreed. This will usually be in the form of documentation outlining a design solution and may be supported by a full set of specifications.

The process itself is highly creative, iterative and interactive, and as such good use can be made of team brainstorming techniques. All the discussion and team activity can be recorded on flip-charts or equivalent so that all can see the direction in which the team is moving. Of particular importance are the unanswered questions or open issues which will arise and assumptions which are made. These should be kept as working lists so that they can be closed or confirmed before the study is be completed.

2.8.1 Management Responsibilities

Architecture concepts can have a significant bearing on the drive to improve quality and efficiency, reduce costs and help meet the increasing requirements for the demonstration of safety conscious design. However, there are barriers to the early architectural design approach being widely adopted. The main problems are with the investment requirements in tools and training and the re-emphasising of responsibilities from traditional component engineering.

The investment requirements can be significant as tools such as CASE do not come cheaply, and staff have not only be trained in their use but also to be given the opportunity to learn on the job. The availability of sufficient computing resources must be addressed. It is difficult to justify such investment (such as the cost of a CASE tool platform) in conventional terms by offsetting against savings, as these are difficult to quantify.

Redeploying personnel in architecture teams is perhaps a more difficult issue than that of investment. Experienced component engineers are, by their very nature, focused on their one particular aspect of a vehicle electrical system. This expertise is recognised by their management and this may lead to an unwillingness to release them for what may be considerable lengths of time to participate in architecture studies.

Architecture core team members must be prepared to commit to the work involved and apply themselves fully to the needs of the team, and a manager responsible for a such a team should make available all the relevant information and demonstrate his support for whatever the team decides. This is a philosophy which must be recognised from senior positions in an organisation, and as such is strongly related to the ideas surrounding Total Quality

Management (TQM). The team should be **the** authority for deciding the electrical architecture and management's role is to support and facilitate the decision process. (See MISRA Report 7-Subcontracting of Automotive Software).

3. Communications and Multiplexing

3.1 Introduction

Multiplexing is the term given to the technique of using the same transmission medium to transfer multiple and separate pieces of information. The medium can be any communications channel but in motor vehicles twisted copper wire is the most likely, with perhaps optical fibre being used extensively at some point in the future.

As the multiple pieces of information are separate, some way must be found of keeping them separate as they go down a single channel. Two basic techniques exist, known as frequency division and time division multiplexing. The frequency division method can be used for analogue or digital signals and is most often found in large broadband communications systems. It is not likely to be used for integration in motor vehicles.

Time division multiplexing is the use of high speed digital sampling to allocate each signal a small time slot in turn; the rate of sampling is high enough to make the signals look continuous. The information is passed in packets of binary data, and can be transmitted along a copper wire or pair of wires. The speed of the binary data transmission in bits per second (BPS) is the main constraining factor on how many signals can be multiplexed on a single channel. The data packets, or messages, are recognised either by a known time reference or by a message header or identifier. In motor vehicle applications the latter is more usual.

The application of multiplex technology can be used to reduce the number of wires in a vehicle, as without it each signal would require its own wire. The SAE (US Society of Automotive Engineers) has categorised the types of message carried by a multiplex network into three classes - A, B & C. This classification is still widely used in papers on multiplexing, although it has been somewhat overtaken by events.

3.2 Vehicle Multiplexing Classes

3.2.1 Class A

Class A refers to the application of multiplexing to switching systems. These can be characterised by low cost, simple components which may not require a programmable element. Their response times are of the order of 10’s of milliseconds, and the speed of the communications channel necessary is anything up to 10 Kbits/sec. An example is the central locking system of a car.

SAE Definition :

> A system whereby vehicle wiring is reduced by the transmission and reception of multiple signals over the same signal bus between nodes that would have been accomplished by individual wires in a conventionally wired vehicle. The nodes used

> to accomplish multiplexed body wiring did not exist in the same or similar form in a conventionally wired vehicle.

3.2.2 Class B

Class B refers to the application of multiplexing to allow the sharing of information between subsystems. This is characterised by moderate complexity control modules which need the same information to carry out their functions, where without multiplexing each would need its own sensor. The response times are of the order of 10's of milliseconds, and the speed of the communications channel necessary is of the order of 100 Kbits/sec. An example is the sharing of engine coolant temperature between the engine management, air conditioning and instrumentation modules.

SAE Definition :

> A system whereby data (e.g. parametric data values) is transferred between nodes to eliminate redundant sensors and other system elements. The nodes in this form of a multiplex system typically already existed as stand-alone modules in a conventionally wired vehicle. A Class B network shall also be capable of performing Class A functions.

3.2.3 Class C

Class C refers to the application of multiplexing to allow control modules to communicate on a real time control basis. The communications channel here is actually part of the control loop and the processing is distributed between the modules. The response times are of the order of milliseconds and the speed of the communications is anything up to 1000 Kbits/sec. An example is the interaction between ABS and Engine Management to provide anti-wheel spin or traction control.

SAE Definition :

> A system whereby high data rate signals typically associated with real time control systems, such as engine controls and anti-lock brakes, are sent over the signal bus to facilitate distributed control and to further reduce vehicle wiring. A Class C network shall also be capable of performing Class A and Class B functions.

In practice it is difficult to envisage 3 separate systems in a single vehicle, and the most likely scenario is to have 2 networks. These would be called Class AB and Class BC systems.

3.2.4 Current and Future Applications

In current production vehicles (1994), applications of multiplexing are fairly limited. In some executive cars, multiplexing is used to facilitate communication between engine, transmission and anti-lock brakes. There is also some application of low-speed body electric multiplexing to overcome problems of harness congestion, and serial communications is used increasingly

with security systems to overcome the problems of "hot wiring" (overcoming the ignition lock by breaking it). Some agricultural vehicles use multiplexing more extensively, albeit at much smaller production volumes than most of the industry.

Many high volume vehicles in the planning stage are being designed with an electrical system which incorporates multiplexing. The information carried by such multiplex systems can be divided into two main groups (Class BC & Class AB):

Class BC:

> Mechanical command and control information: Vehicle state (speed, engine speed, temperatures, transmission state, etc.)

Class AB:

> Driver & passenger information: In-car entertainment information, car telephone, heater/aircon, body electric switching & vehicle state information.

Because of the widely differing requirements of systems to encompass both sets of information, some manufacturers are opting for two separate multiplex buses, one for real-time control and one for driver/passenger services.

3.3 Currently Available Protocols

A number of multiplexing systems are available; for each there must be a clearly defined set of rules or a protocol standard to which all nodes comply in order to ensure that communication works.

The standardisation of protocols is an important issue, otherwise it would be impossible to guarantee compatibility between control modules which are supposed to communicate. Standardisation offers the industry considerable economic advantages in that the overall volumes of multiplex chips that operate to industry standard protocols are highly attractive to the multinational semiconductor suppliers, and they are incorporating these devices into their range of standard products.

Standards fall into three basic categories: international, company and proprietary, the application of which have particular advantages depending on the operational requirements.

All the main internationally used protocols so far follow the same basic scheme for access to the bus. They do not have a master node that controls communication, but rather any node can gain access to the bus when it is free. The node that gains access is the node which sends the message with highest priority. Priority is determined during the transmission of the message header. This is organised by biasing the hardware such that a low level on the bus and a high level have unequal drive strengths. One is dominant and the other recessive. If two nodes both transmit simultaneously a bit of opposite sense, the bus takes the status of the

dominant level. This provides the means of arbitrating between nodes competing to transmit simultaneously. If both nodes start to transmit the message identifier simultaneously, the first one to transmit a dominant level whilst the other transmits a recessive level wins arbitration and the node which lost must cease transmitting again until the bus is next free. This scheme is known as Carrier Sense Multiple Access with Non-Destructive Arbitration (CSMA+NDA). CAN, VAN and J1850 all use this scheme, although their individual schemes to transmit at bit level are quite different.

There is an alternative strategy called a token ring, in which a token is passed from one node to another, and the only node with permission to transmit is the one in possession of the token. (see [6] for a discussion of the relative merits.). Other strategies such as polling do not work on a multi-master bus.

Currently all three international standard protocols recommend the use of unscreened twisted pair copper wires as the transmission medium. The indications are that this method provides good immunity to EMC with the present design of vehicles and current data rates (up to 1Mbit/sec). The EMC performance of such a vehicle would need to be examined carefully at the outset to test these basic assumptions. (see MISRA ST3-Noise, EMC & Real Time and [7])

3.3.1 International Protocol Standards

CAN ISO/DIS 11898. This was originally designed by Bosch and Intel although now it has full Draft International Standard status for Class C high speed communications within the ISO organisation. The standardisation of a lower speed Class A application of the same protocol is subject to separate ISO activity ISO/DIS 11519, which also incorporates VAN. It is currently known to be of interest to Mercedes-Benz (in production S-class), BMW, Porsche and Audi in Germany, Jaguar and Rover in the UK, all truck and bus applications in the US, as well as in many diverse non-automotive areas such as marine, process control, security, lifts and even fruit machines! CAN is already in volume production on a number of vehicles. Some of the earliest applications have been agricultural equipment.

ISO 9141 and ISO 9141-CARB. This is primarily a diagnostic protocol used exclusively for point to point communication between the vehicle and an external service tool. It has been widely adopted to meet the legislative requirements in the USA for OBD I and II where fault codes must be stored by the vehicle and reported via the communications link to a service test tool. It is a low cost solution often only requiring a few additional discrete components. Most European manufacturers currently use ISO 9141 for their diagnostic applications. ISO 9141 is not intended for communications within the vehicle.

SAE J1850. This is an integration of the recommended practices for Class B systems incorporating the three different standards of the "Big 3" US automotive companies Ford, GM and Chrysler. They are however regarded as international because of the size of the companies which operate globally. J1850 protocols are being used by the "Big 3" in the US and considered in Europe for class B multiplex applications. They are also mandated for OBD II for diagnostics in US legislation, by both EPA (Environmental Protection Agency) and

CARB (Californian Air Resources Board), as an alternative to ISO 9141. A serious drawback to J1850 is that Ford and General Motors have developed the basic J1850 protocol into company standards that are incompatible.

VAN (and CAN) ISO/DIS 11519. VAN stands for vehicle area network. This is a French technology developed by Renault-Citroen-Peugeot similar to CAN and is being used by their motor industry mainly in class A and B multiplex applications, and the intention is to expand to class C. It is being well supported by most of the major silicon manufacturers, and is covered by an ISO international standard. It is mainly used within France, however both Volvo in Sweden and Fiat in Italy are known to be investigating it.

3.3.2 Company Standards

As the international standards scene is only just stabilising, most companies with a need to introduce some form of communications, developed their own approach. The Japanese are the main players who still advocate the advantages of their own protocols (e.g. Mazda's "PALMNET") although they are gradually aligning themselves internationally and Volkswagen in Germany are pursuing the A-BUS protocol.

The Americans were able to pool their ideas through the SAE and that led to the J1850 "Standard", which incorporates Ford's SCP (Standard Corporate Protocol), GM's DLCS (Data Link Controller Serial) and Chrysler's PCI (Programmable Communications Interface) by allowing options for things such as frame format, CRC's and physical medium. Whilst each of the three protocols meet the letter of the "standard" they are incompatible. However within each company the individual protocols are well defined and understood, and because of the size of the US auto market, each company can support its own protocol.

The major disadvantage associated with setting a company standard is the availability of silicon to support it. The company will either have to be responsible for fabricating its own devices or be prepared to pay one of the semiconductor manufacturers to do it. This can be an expensive and risky business.

As an alternative for the lower speed systems, it is possible to support some protocols in software with a suitable microcontroller, and some specialised micros are being developed that are optimised for this task.

3.3.3 Proprietary Protocols

These are similar to company standards except they are designed with a specific application in mind, where some special attribute such as cost, speed or integrity dominates. Vehicle manufacturers will need to assure themselves when they use such multiplex systems that they are satisfactory. All aspects of the design of such a system need to be examined and understood, possibly by external consultants, and its performance carefully evaluated to ensure that it is indeed "fit for purpose".

There is a further group of protocols, used entirely within a single module, for communication between different parts of a circuit. Examples are chips that intercommunicate via an I^2C bus, or the interconnection between two PCBs. The vehicle maker should ensure that the component supplier which has used such protocols has produced a sound design which can be relied on to operate correctly in the automotive environment.

It is worth noting that the recommendations of MISRA on this subject apply to all these protocols where appropriate.

3.4 Protocol Selection Criteria

As there are a number of options for which protocol to select for the development of an integrated vehicle electrical system, it is important to have a rationale for justifying the choice made. This section lists some of the points which should be considered.

What Data is to be Carried on the Network?

For what purpose will the data be used? For example, road speed information can be used to control a cruise control or the delay on an intermittent wiper.

Application

Is the protocol matched to the application, such that protocol class A, B or C meets application class A, B or C? Note that a Class C protocol can also do A and B application and Class B protocols can also do class A applications.

Speed

Is the speed capability of the protocol high enough? i.e. Is the bit rate in kbits/sec sufficient to meet the data rate requirements of the system? This must include allowance for delay or latency introduced by the communications medium, under maximum traffic. More expensive hardware is generally required for higher speeds. In this respect, the bit representation of CAN offers more efficient use of available bandwidth than J1850 or VAN (see [8-9]).

Capacity

Does the protocol support enough message identifiers to carry all the information required by the system?

Robustness

Is the error detection and handling capability robust enough to support the integrity requirements of the system? i.e. Hamming distances, residual error rates, number of diverse mechanisms for error detection, etc. (See [10]).

Availability

Is the protocol widely supported by silicon? If so is there a range of devices available, perhaps from different silicon suppliers. It may be possible to pick the feature level offered by different devices to different ECU's, all of which still meet the same communications standard. This optimises cost. Which family of devices is most likely to be supported long term? The motor industry requires support over a much longer timescale than the computer industry.

Software Drivers

Is the protocol heavy or light on software to receive and transmit data? This will affect the software design complexity to support the communications. For some chips standard software drivers are available.

Cost

Is the cost of equipping each ECU with the communications facility able to bring overall system cost benefits? Note this can often mean that piece prices increase on some ECU's but overall costs are reduced due to savings in other areas such as harnessing and fusing, and considerable benefits in terms of enhanced functionality can usually be realised.

Flexibility and Expandability

Does the protocol offer the potential for sufficient expansion and ease of modification throughout the life of the product? After all this is one of the major potential benefits of introducing communications at all.

There may be other criteria which are important for specific reasons, but those listed above are perhaps the most relevant for selecting one protocol for an application.

In practice, international politics and national / regional rivalries are unfortunately likely to be key considerations. At present there is a battle going on between the multiplex protocols that has parallels with the VHS/Betamax video format wars of years gone by.

3.5 Specifying Automotive Communications

Communications systems have for some time been specified in relation to the generic ISO standard 7 layer model. This covers all communications system requirements from mobile telephone to global wide-area-networks. The layers are :

1 - Physical

The part of the system covering the transmission medium, transceiver circuits and electrical parameters of the communications channel.

2 - Data Link

This includes requirements such as coding and bit level error detection/correction, in fact anything which contributes to point-to-point error free communication.

3 - Network

This handles items such as the routing of data across a multi-node network, such that it looks like point-to-point even though there may be many individual connections.

4 - Transport

This manages features such as failure recovery, if one of the node to node links should develop a fault, and re-establishes an alternative link while maintaining connection at this level.

5 - Session

This manages the process of logging on and off a system connected to the network from remote terminals.

6 - Application

This layer contains the services and features actually available on the network

7 - Presentation

This is the part which offers the information generated as a result of the application which has been across the network to the user, e.g.. displays on a screen.

Most of these layers do not apply to automotive applications. For multiplexing only layers 1 and 6 apply plus one other which condenses layers 2-5. Other applications may need reference to other combinations of the possible 7 layers, for example security applications and communications between vehicle and diagnostic tool may need a log-on/off facility and therefore a Session Layer.

3.5.1 Three Layer Model

This is a condensed version of the ISO-OSI 7 layer model which is more appropriate for real-time applications such as automotive multiplexing. The layers are physical, transfer and application. See Figure 3.

3.5.2 Physical Layer

This is the same as ISO model for the wires (or perhaps optical fibres), connectors, transceiver circuitry and electrical parameters relating to the communications bus. Most of this layer is covered by the standards and the specifications of the communications chips themselves.

3.5.3 Transfer Layer

This is a new layer which effectively contains all the relevant functions from the traditional data-link, network, session and transport layers above. The reason for combining them in automotive applications is that many of the situations covered by the ISO seven layer model do not apply to automotive networks. Items such as routing, logging on, invisible failure recovery do not apply to automotive in-vehicle networks.

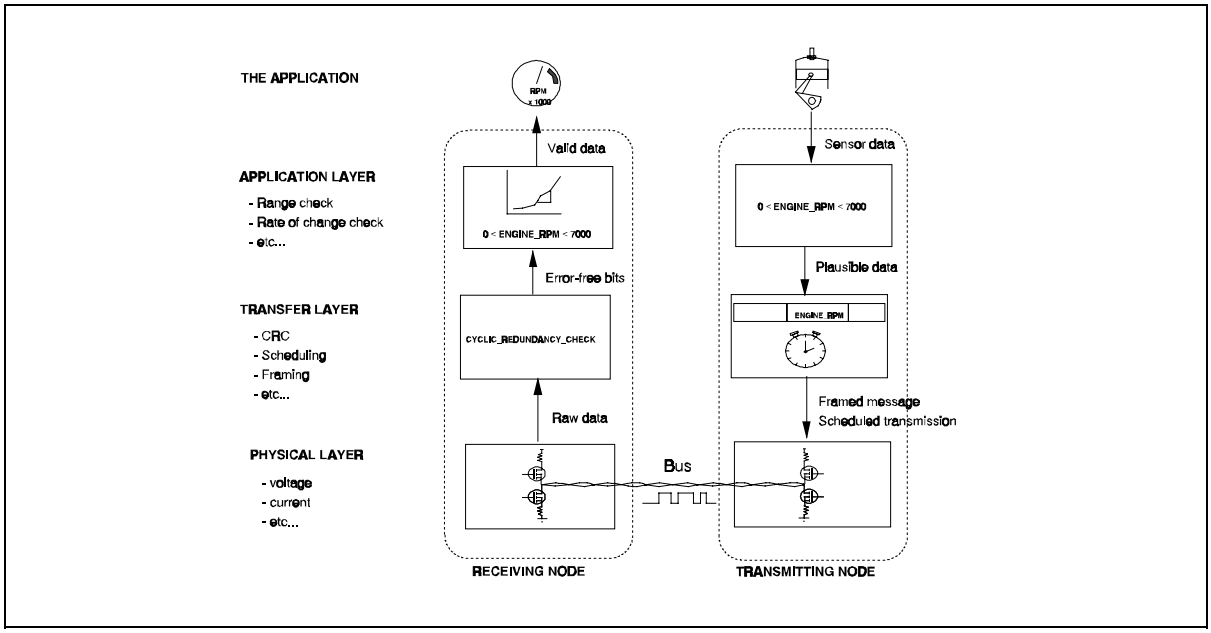


Figure 3 - Three Layer Communications

Messages and message packet are the main subjects of the transfer layer and the specification of this layer is the key to a reliable and robust network. It is mainly carried out by the driver software and therefore should attract the attention of MISRA. Transfer layer functions are

- Message transmission scheduling
- Message to/from packet conversion
- Reporting of timeouts on parameters (each parameter may have a different repetition rate and hence timeout)
- Detect and signal network failure to the application
- Handle network startup and synchronisation

It is useful to think of the transfer layer as being made up of the elements shown in the diagram in Figure 4.

3.5.4 Message Scheduling

Scheduling of message transmissions is an important part of the network to ensure that the correct data update rate or cycle times can be achieved. It is considered bad programming practice to have the transmission drivers fragmented and distributed within the application code, it is much better to have a centralised scheduling algorithm based on one timing reference.

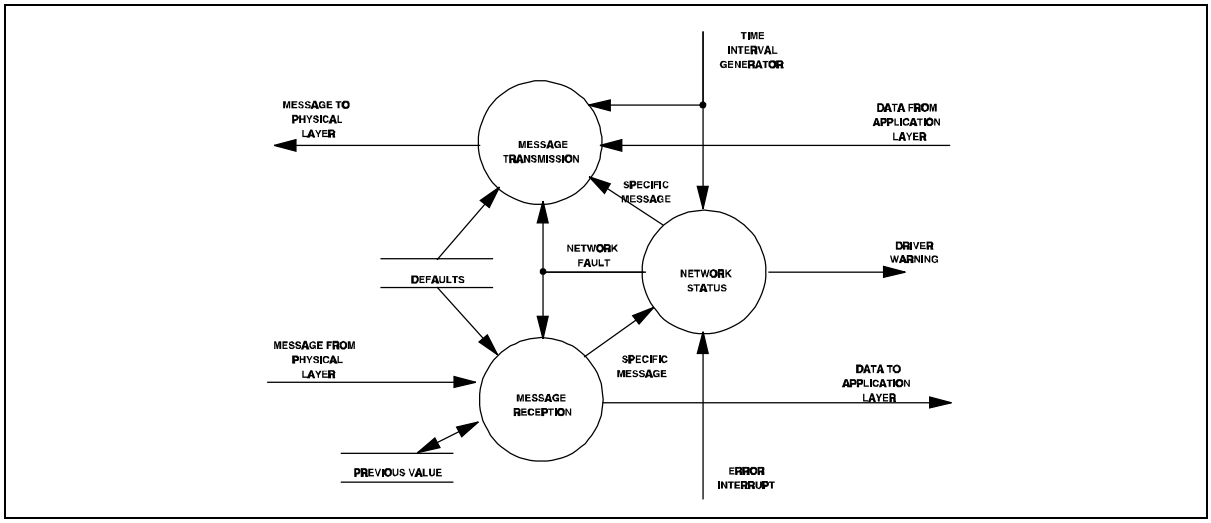


Figure 4 - Transfer Layer Structure

Since vehicle control systems must be very robust, a major goal of a communications network must be to ensure that it operates in a predictable manner. This means that the worst case timing behaviour of message transfer can be determined, and that there is sufficiently high confidence that this is correct. Simulation is often the approach taken to achieve this. If a low bus loading is specified the confidence in the simulation results is high. However, at higher bus loading a smarter approach is required.

The foundation of real time systems engineering is scheduling: this is because there is likely to be a number of competing resources (ie. message transmissions), and each of these will have a different time deadline.

Most of the common multiplex protocols use so-called priority pre-emptive scheduling but with the additional feature that once a message has started transmissions it continues until complete. This basically means that the messages in the queue are transmitted in order of priority by an arbitration process and the priority is set by the message identifier.

Schedulability analysis can be used to compute the worst case latency (or transfer delay) for a given message. In order for the analysis to hold, several assumptions are made: The maximum message delay should be less than minimum cycle time for that message. The messages should be assigned worst-case latencies and will not over-run these.

The aim when writing software drivers for multiplex protocols is to implement a priority queue of messages. This should be based on the message deadlines and sufficient mechanisms should be designed to ensure that each message cannot violate the assumptions of the schedulability analysis. Note that a highly critical message may have a long deadline, whereas a low criticality messages may have a short deadline. When assigning priorities, the deadlines and the criticalities must be considered. Having the ability to adjust messages identifiers and hence priorities gives advantages for scheduling bus performance.

Analysis tools can be used to analyze and configure the system to meet the bus loading and message deadlines. Instead of over-engineering message transmissions rates to allow for errors, it is possible to build an error estimation function which can predict the upper bound limit on message overheads.

3.5.5 Network Status

This area contains the functionality for monitoring the status of the network, primarily to check that all the nodes which should be able to communicate can indeed communicate. This can be done by periodically checking the reception of specific messages from each node, and providing a regular message for them to check. If the lowest priority identifiers are used for the network status messages then any clog-up caused by erroneous saturation transmission of other messages will also override the status messages and hence be detected. The rate at which the network status messages are checked will depend on the integrity and time response of the network.

It may also be necessary to handle the error warning facilities which are often available from the hardware in the form of an interrupt. It may be deemed necessary to warn the driver of any network failure detected here.

3.5.6 Defaults

Each communicating node should have a permanent store of default values for data which is both transmitted and received by the communications network. These default values should always ensure that the vehicle is left in a safe state. Hence if for example a sensor fails and normally its data was to be transmitted, instead a default can be used. Similarly, if a node fails such that another node is unable to receive any data from it, the default values can be passed into the application layer.

It is usually important that the application layer knows when a default value is to be used, as it may have additional information available that will improve its estimate of default value. (For example engine speed can often be estimated from road speed and gear status.) It may therefore be more practical to confine the actual data to be used in failure modes to messages that are interpreted as commands to the relevant application layers to use default values. A fail message should be sent across the network rather than default values, and the relevant application layers should choose suitable defaults.

3.5.7 Application Layer

This layer is almost the same as in the ISO model, as this is where the data acquires meaning and its use within the system becomes relevant. Here the general requirements for software reliability apply, and no specific recommendations are added relating to communications. Applications layer functions are:

- Handle the validation / default substitution algorithm for each parameter on the basis of the data/status information provided by the transfer layer.

- Perform the primary function of the ECU.

- Provide data/status to the transfer layer. The status will indicate to receiving nodes if this node considers the data to be good or bad.

The application layer consists of the functions that generate or act on the data packet. It is only at this level that the data assumes any meaning.

It is in this layer that plausibility checks need to be applied to received data. These may be specified for application to specific data depending on the integrity requirements of the function it is used in.
For example :

- Range Checking an item of data before passing it to the application; if it is out-of-range then ignore it and use the previous, default or an inferred.

- Plausibility Checking an item of data for rate of change limitations or illegal states; if an error is suspected then use the previous, default or an inferred value.

3.6 Design Aspects of a Vehicle Multiplex System

A vehicle multiplex system will normally be used to link a number of electronic control modules throughout the vehicle. As such a number of different specialist component suppliers will be providing software which will interact with the network by supplying and receiving data to and from it. The suppliers will in general be specialists in the design of their own subsystem, and even if they all work for the same component supply company, it is most unlikely that they will all be from the same division of that company. A more likely scenario is that each module attached to the network will be supplied by a different company, some probably even located on different continents.

The vehicle maker with overall system responsibility needs to maintain close control and communication between the various design teams that will be using the multiplex bus. The vehicle makers may like to consider sub-contracting this specialist task if they do not feel that they have the necessary skills in-house.

In designing a vehicle which will rely on multiplexing, careful consideration needs to be given to the project management of the development of the multiplex system. Component suppliers will have to work together rather more closely than hitherto, and have to accept constraints on their software design, an area in which they have traditionally had considerable freedom.

3.6.1 System Partitioning

This activity [see section 2.2] will include listing the functions in the vehicle that require electronic control and assigning them to particular modules for which individual component/system suppliers can bid to design and supply.

An important consideration of system partitioning is the physical routing of the multiplex cabling and location of the various modules. If the network is going to carry data that will make the vehicle undriveable if the network fails, the whole vehicle will be made rather vulnerable to minor knocks if the network is routed to the periphery such as into the doors or down to the lamp clusters. It is better to keep the cabling for such a network inside the core of the vehicle. If signalling to doors and lamps is required, this can be achieved with either buffered spurs which can be isolated if a short occurs in them, or with a separate network.

At this stage, an approximate list of the inputs and outputs for each module needs to be made. Included in this list are the data items to be input/output by each module on a multiplex bus.

From this can come some basic decisions about the communications structure around the vehicle, as follows:

- How many networks will be used?
- What modules will link into each network?
- What is the maximum permissible latency time for the high priority messages on each network?
- What is the maximum data rate required for each network?
- What method will the diagnostics use to interrogate the control modules?

3.6.2 Specifications

The parameters such as bus speed derived from the above analysis should be specified in the detailed requirements submitted to each supplier for the electronic modules that will attach to each network. Where the interface circuit is not covered by the standard, the car maker may wish to provide a recommended circuit to the network to ensure that component suppliers maintain compatibility with each other.

The functional requirements should be given in as much detail as possible, together with the amount of memory to be reserved for diagnostics (assuming the diagnostic specification is incomplete).

3.6.3 Choice of Suppliers

When choosing suppliers of electronic modules, cost is usually the overriding factor. The following are some specific criteria to be evaluated in relation to design components that interact with a multiplexed bus. In order to compare rival offers, it is important to ensure that:

- Each supplier has allowed sufficient time to develop the software, especially the network software.
- Each chosen microcontroller has enough spare resources to be able to be expanded, or a larger version exists that can be substituted with minimum program rewriting.
- Each supplier agrees to co-operate with each other. (see MISRA Report 7- Subcontracting of Automotive Software).
- A responsible authority has been appointed (probably the company responsible for the overall network), to define the scheduling algorithms for multiplex messages.

3.6.4 Operation of the Network

The main consideration is the integrity and stability of the network. It must operate reliably and be designed to avoid situations in which it fails because of overloading, or essential control processes are delayed because of deadlocking or live-locking.

Assuming that the network follows a CSMA scheme (such as CAN, VAN or J1850), it is suggested that all the message identifiers are stored in non-volatile memory in all the electronic modules. By this means, the priority of any message can be adjusted if there is a concern with its operation.

Messages should not be transmitted more often than required as this simply uses up surplus bus bandwidth and receiver resource.

All receivers of information from the bus should perform additional checks on data integrity at application layer. Is the data within preset limits, and is it reasonable? Many variables can only change relatively slowly, and software should reject data that seems to show impossibly rapid changes.

All nodes on the bus should have a fall-back mode of operation if the data they need from the bus to operate fully is unavailable. This can either be estimated from other information available to the module, or be given a default value.

The performance of the individual modules needs to be evaluated in this bus-failed mode to check that the vehicle remains safe and legal.

3.6.5 Message Schemes

One method of minimising the probability of bus failure due to overload is to assign a time slot to certain basic information that has to be shared around the network. Such data might include road and engine speed, various temperatures and the status of certain body systems (lighting, wipers, etc.).

Status commands or messages should be sent at regular time slots, even though their status may not have changed from the previous transmission. In the unlikely event of corruption of a message, or a message being incorrectly interpreted at a receiver, it reduces the time before a valid message becomes available.

A particular advantage of sending messages at regular time-slots is that calculation of rate of change for plausibility checking purposes is simplified. The receiving node may not need to store the time at which such fixed-period messages are received in order to calculate rates of change between successive messages.

For critical data it may also be worth considering augmenting the basic protocol with other features to increase the confidence in its correctness. For example three techniques are : "oversampling" or sending the message at a higher rate than required will allow the multiple receptions to be compared and voted on; using the data field to provide more bits of cyclic redundancy check, or even extending it to error correction codes; or time tagging each message so that its validity at any point in time can be verified before it is used.

3.6.6 Bus Integrity

One node should be given the task of network manager. It should ensure that all nodes on the network are sending messages correctly, may involve sending dummy messages to check the status of nodes that do not transmit regularly. For extra security, another node should also be given the task of checking on the network manager node. If faults are discovered, a back-up strategy should be defined, which will depend on the importance and nature of the data supplied by the missing node. It will probably consist of sending a warning message to the driver and other nodes.

The network manager node can also be used to look for "illegal" messages on the bus, ones from nodes that have been added as an unofficial modification. This information can be stored and read via the diagnostic system as a defence against tampering and product liability. It should be possible to allow extra nodes to be added legally to the vehicle via an official service outlet. In this case, the network manager’s non-volatile memory will have to be loaded with the identifiers from the newly added node. This could be done by means of the diagnostic facility.

3.6.7 Bus Loading

The bus loading (the mean proportion of the time for which the bus is active) will determine the worst case propagation delay for any particular message. The higher the bus loading, the

more efficiently the bus is used, but the greater the worst case propagation delay. With CSMA systems, high priority messages are largely unaffected by bus loading, but above a critical bus loading, lower priority messages suffer greatly increased propagation delays. Careful analysis is needed of the input data required by each process to ensure that all data, especially low priority data, required by every process always remains valid. Especial care is needed of processes whose outputs are critical.

3.6.8 Diagnostics

There are two philosophies on diagnostics. The view in the USA has tended to be that a diagnostic tester can be plugged into the bus directly via a connector located somewhere in the vehicle to talk to the various control modules directly to perform calibration checks, obtain status reports and data stored in their non-volatile memories such as records of intermittent faults. Indeed this is the scheme embodied in the Californian CARB legislation.

The view more prevalent in Europe is that the bus is likely to carry essential control information whose propagation delay has been carefully calculated based on known worst-case network traffic. To allow third party diagnostic equipment direct access to the bus will put an additional unknown into this calculation. It is better to use an ISO 9141 connection to one of the modules on the bus which acts as a gateway to the main network.

The vehicle maker will need to evaluate the risks and costs associated with each diagnostic philosophy and decide the appropriate strategy.

4. On-Board Diagnostics

4.1 Introduction

The main objective of on-board diagnostics is to seek to maintain the vehicle in a safe and legal state.

This is addressed by detecting errors and faults as soon as they occur, taking corrective action - possibly setting a default state - and warning the driver of the occurrence of the fault. For many faults, it is possible to initiate a default state, or take corrective action which will result in largely unaffected driveability; for this reason, it is necessary to warn the driver in such a way as to indicate the seriousness of the fault. This is particularly important where there might be an effect on the integrity of the gaseous emissions, for example, since this may render a vehicle illegal to drive in some markets (e.g. CARB OBD II).

A secondary purpose is to store data which assists repair personnel to identify the nature of the fault: fault or trouble codes. These may be read by an off-board diagnostic tool, or more simply, by making a connection which initiates flashing of the fault warning lamp. Although the latter is an important feature of on-board diagnostics, it has significant limitations if used in isolation.

Diagnostics may play quite an important part in assuring customer satisfaction in the event of trouble, by speeding diagnosis and rectification of the fault, as well as keeping the customer mobile, safe, and legal for as long as possible.

Diagnostics should not be added on as an afterthought: they must, for best effect, and maintaining integrity, be included as part of the overall system design strategy. It should be clear what the objective of the diagnostic scheme is. For example, an objective of some diagnostic schemes is as an arbitrator for warranty claims; for others, safety alone is the main objective. It should also be clear whether there is an objective to aid in roadside repair, or restrict access to franchised dealers' personnel.

A fault code alone may be misleading; for example, a MAP sensor will give an erroneous reading, in a closed-loop system, for a variety of reasons, which may relate to mechanical as well as electronic problems. It is usual for off-board tools to carry out very detailed interrogation of an ECU (electronic control unit) displaying a fault; quite often, a sequence of events provokes a fault code being stored, and it is this sequence which provokes the correct diagnosis to be attained. When establishing a diagnostics strategy, the limitations of on-board schemes should be carefully considered.

4.2 Fault Detection Strategies

4.2.1 Detection and Management in Software

In principle, software can be used to detect three kinds of fault:

- Sensor signals non-existent or incorrect (out-of-range, rate of change, checked against idealised internal model, etc.)
- Actuators not performing as intended (non-operational or restricted)
- Processes within the host system not functioning as specified, where the malfunctions are not due to the processor itself (e.g. endless loop, misinterpreted op-code)

4.2.2 Sensors and Actuators

Unless a sensor has an internal fault-reporting scheme, which communicates a separate flag to the host system, fault identification can only be carried out on the signal-conditioned and digitised data presented to the host system processor. Thus it is inherent that fault reporting for any sensor system will usually include faults in the signal-conditioning circuitry for the sensor(s) in its domain. It is thus usually impossible to differentiate between faults due to the sensor itself, and faulty signal-conditioning. In general, sensor failure is relatively easy to detect; the sensor will either fail high (to supply), low (to ground), or cease to respond to changing mechanical quantities.

In the case of actuators, a fault may be detected by inference by examination of the behaviour of the object being controlled by the actuator (sensor signals, versus expected condition), or by a fault-reporting mechanism in the actuator itself. For example, it is possible to check solenoid inductance by applying a step voltage and measuring back EMF, or electrical resistance can be monitored by measuring current. There may be sufficient sensors to be able to make a valid decision regarding demand versus actual state. Care must be taken, however, in interpretation; for example, say, a throttle demand does not result in the expected engine output, it could be erroneous to conclude that the butterfly had not opened to the position demanded. The engine might be short of fuel, parking brake on, etc,, etc.

In closed loop systems on-board condition monitoring can be employed to raise system integrity by trapping failures before they occur. This may be necessary because as sensors and actuators age and wear the performance of the system will change. This is the basis for OBD II for engine emissions. Several methods exist for doing this, for example :

- comparing directly sensed parameters with inferred values from other sensors
- using redundant sensors
- compare the system responses against those of an internal model

- stimulate the system by controlled modulation of actuators during appropriate periods and measure the responses.

To employ condition monitoring successfully requires the careful setting of thresholds for the system components and a detailed understanding of the variation that exists in any mass production environment. It is reasonable to assume that if system condition monitoring is used to ensure a system stays within a defined operating range in order to avoid a particular safety hazard, then the degree to which the monitoring is engineered will depend on the severity of the potential hazard, i.e. the higher the integrity requirements the more comprehensive the condition monitoring. [see MISRA Report 4-Software in Control Systems]

4.2.3 Process Malfunction Detection

Processes may fail because of:

- the influence of noise/ EMC
- communications failure
- incorrect design of software (out-of-range handling, interrupt handling, etc.)
- faulty processor action

The first three may be detected by software. The last one may usually only be detected correctly by a hardware watchdog or a voting arrangement.

Usually, the effect of noise/ EMC will be to corrupt a sensor value, or a data conversion value. This will be perceived by a software error capture routine as a sensor problem. Often, error capture routines are designed to record a fault code history, and may be used to invoke a default state. Error capture routines may compare incoming data with the "expected" range in a look-up table; may look for an out-of-range value from a sensor, or a response from an actuator which is not compliant with demand. They also may look at timing of functions in much the same way as a hardware watchdog.

When an error is captured it can be "weighted" so that the response from the error capture routine is related to the number of occurrences, criticality, or longevity of a captured error. Fault codes may not be stored unless the weighting parameters are exceeded. A different weighting may be applied to the invoking of default states.

Certain categories of error may be extremely important, notably for example, relating to dc supply conditions in critical ECUs. These may not be used to invoke a warning light for the driver, but will be accessible to trained staff at the vehicle dealership. A typical purpose of this approach could be to indicate tampering with the ECU, or unintentional corruption of permanent memory contents.

4.3 Criteria for On-board Diagnostics

If a detection strategy is to be effective it must satisfy the requirements that:

- it detects only "genuine" faults
- it invokes limp-home states in a safe manner
- it provides information to the driver in a reasonable fashion
- it stores the fault information and makes it accessible to authorised repair personnel.

For these requirements to be met, it is essential to clearly lay down and understand the criteria for each part of the detection and diagnosis process.

Detected faults must invoke one or more of a range of actions:

- warn the driver/ not warn the driver?
- store fault data till reset?
- store fault data semi-permanently?
- set limp-home mode semi-permanently?
- set local default states?

Faults may be minor, intermittent, serious involving risk to plant or serious involving risk to people. Fault detection strategy should be hierarchical, with action/ reaction cross-referenced to the seriousness of the fault detected. It may be considered that some faults, unless they occur for more than a specified period of time, or more than a specified number of times, should be ignored. More serious faults will normally invoke a default mechanism, and possibly a limp-home state.

4.3.1 At Start-up

The condition of each sensor and (where feasible) each transducer should be interrogated prior to the activation of the starter. Sensors may be checked for continuity; resistance within a predefined range; response to a defined electrical offset. Transducers may be checked for continuity; resistance between predefined limits; and possibly inductance between defined limits (by virtue of the back EMF measured for a step voltage applied.)

4.3.2 During Operation

Actions depend on the type of system being considered.

For engine management systems, sensor voltages are monitored when sensors are addressed in the normal course of operation, and if any are found to be out of range, a fault may be assumed. Actuators cannot generally be tested during system operation.

For systems such as ABS, which have only occasional operational demands, sensors may be monitored continuously during idle periods. Actuators may be monitored for continuity only, once the vehicle is in service. A simple test to check wheel speed sensors is to compare each wheel speed sensor frequency with the value expected from the vehicle speed.

For an air bag system, the firing mechanism is interrogated by pulsing with a small voltage, and measuring current. The measured current must remain within an acceptable range.

4.3.3 Invoking Limp-home State

Depending on the seriousness and criticality of a fault, a limp-home state may be invoked to allow the vehicle to continue its journey in a safe manner but with some form of performance or functional restriction. An engine management system for example, may be designed so as to keep the engine running but with a deterioration in driveability e.g. by limiting engine rpm. Limp-home should mean just that, as opposed to an error accommodation scheme. It should aim to maintain as much functionality as possible and not unnecessarily degrade driveability to promote a more urgent repair action.

Limp-home should only be invoked when there is no other alternative, and if it is going to significantly impact on the driver’s control of the vehicle, adequate notice of the event should be given, as far as this is possible.

4.3.4 Warnings

Consideration should be given to the mechanism of warnings, so that the driver is not unduly alarmed, or unnecessarily warned for minor faults. Such over-warning can lead to confusion or a nonchalant attitude towards heeding the warning.

The decision process for invoking both warnings and limp-home states should be robust and based on a hierarchical structure which ensures that minimum reaction is implemented whilst containing and compensating for the fault detected.

The most common warning method is a lamp on the fascia. This has two disadvantages:

- it gives no information about seriousness
- it will be one among a possible plethora of warning lamps in a high specification vehicle.

A "generic" warning lamp may be used, supported by a further method of indicating to the driver which system is at fault, how serious, etc. This may be in the form of text, or

synthesised speech, or symbology. Standard symbols are specified for most vehicle warning and information functions.

If a body or diagnostics computer is provided in the vehicle, maybe as part of the trip computer, actual fault codes (alphanumeric, if required) can be displayed. However, currently such approaches are very restricted in their content, and are usually unavailable to the driver as they would only lead to confusion.

Text may be an attractive option. High-specification vehicles, especially in North America, are often fitted with an information display, which may be used to display warnings, control positions, display navigational, trip and other information. Some vehicles use a cathode ray screen, in front of which is a touch screen, to enable controls to be effected directly from driver interaction with the display. Alternatively, a vacuum fluorescent or liquid crystal dot-matrix display may be used. Conventionally, such displays are used in European vehicles for trip computer information. A few vehicles employ independent warning displays, and, possibly, also speech processors.

The use of text or speech processors raises the question of language. There are now well-established methods of displaying different alphabetic systems: Cyrillic, Arabic, Katakana, Hebrew, may all be displayed on a character matrix based on a minimum of 10 by 6 dots. Text may be held in ROM, and the correct language set by jumper links, or set by a test computer, on the production line, or even selection by the driver is possible, in more expensive vehicles.

The same is also true of speech processors. These have been poorly received by the general public in the past, largely because of the way in which they have been implemented: typically, a disembodied voice nagging the driver to carry out a simple task; bing-bong signals; and repetition which can not be stopped do not engender acceptance. There may, however, be a place for some form of spoken warning in future vehicles, if complexity of fascia layouts continues to grow, and also if navigational aids become widespread. A voice warning gives far greater possibility for passing information to the driver without forcing attention to be distracted from the task of piloting the vehicle than any textual or warning lamp system.

A warning system could ultimately be interactive with the kind of system envisaged in the European "DRIVE" programme, with information transmitted from the vehicle to a central computing facility invoking instructions to be transmitted back to the driver of the vehicle using the road traffic informatics interface. This would obviate the need for a large amount of on-board processing to be tied up in providing extensive textual or speech coverage in a multi-language format.

4.3.5 Intermittency and Fault Logging

It has been widely perceived in the US that computer control of vehicle systems results in a large number of reportedly intermittent faults. Often these evade logging by the victim system ECU.

One possible solution to this is to build in a data logger system. This is a particularly attractive option when a body computer or multiplex system is used in the vehicle. This type of data logger is already envisaged by the EC DRIVE research programme (see Section 6.5.1) for recording data relevant to collision situations. In this case, the system would work like a solid-state flight recorder, maintaining a rolling record of the last fifteen seconds of relevant data. Clearly, this approach can be extended to monitoring vehicle systems. There will clearly be stringent speed requirements for the successful capture of a large amount of data.

Multiplex systems have a significant advantage over conventional independent systems in this context, especially if using a high speed bus such as CAN. Set against this must be the constraint that a bus failure could invalidate or incapacitate the process of acquiring data over the multiplex bus, and the restrictions required to bring the bus out to the diagnostic connector.

Faults may be logged in various ways. The most widely used technique is a combination of codes held in RAM which may be cleared on switch off, plus codes relating to more serious faults held in battery-backed RAM or EEROM. Some current systems allow these to be cleared by disconnecting the vehicle battery (outlawed by OBD II), or by using an off-board diagnostic tool. Commonly, the first method will hold a register of occurrences during a period of service of the vehicle, and after a set number, will cause a transfer of the fault data to battery-backed RAM. This may be accompanied by setting a default state, or limp-home mode. It will usually be accompanied by setting the warning signal, even if the earlier occurrences did not invoke this action.

More serious faults, even if not continuously present, may invoke both the warning signal and a default state or limp-home mode. Some lesser faults may also, if they are intermittent and appear more than a preset number of times in a given period, cause a permanent fault code to be stored, invoke a limp-home state, and invoke a warning signal for the driver. At very least, the permanent storage option may be invoked, to alert a dealership technician to the existence of a problem.

Fault logging may be handled locally by the host ECU; in a multiplexed system vehicle, it may be communicated to another host for storage, or even interpretation, default control, etc. Some vehicles also employ "body computers" which collate all fault data, compute service necessity, contain communications to off-board tools, and so on.

4.3.6 Off-Board Support

Currently fault diagnostics are commonly achieved by the use of flash or blink codes where, following an initialisation procedure, the fascia warning lamp is made to flash a particular number of times on starting the engine. The codes are usually flashed as two digits, repeated at intervals, which may be compared with data charts for their interpretation. This has the shortcoming that a reasonable understanding of the system being interrogated is necessary for effective fault-finding.

More sophisticated off-board tools use communication direct with the ECU. The code for supporting off-board diagnostic tools which communicate with on-board systems must do four jobs:

- transmit raw sensor data on demand
- transmit trouble codes on demand
- allow adjustment of on-board parameters by the diagnostic tool
- provide a sufficient level protection against illegal access

The code is usually one of a number of routines in the ECU which are polled during normal operation of the vehicle; when polled, if the diagnostic tool is present, the routine becomes active in communicating with it, and if the tool requests it, influencing the control loop. It is not disabled when the off-board tool is not present.

As yet there is no standard fault code list in Europe (though OBDII requires this in the US - see section 6.6), but work is progressing in this direction, and a document already exists from which message implementations are now being achieved on J1850, ISO, and CAN/ VAN in Europe (KWP2000).

For OBD II the routines and codes which must be present in an engine management system are all laid down in CARB and EPA documents, including the off-board tool communications routine, along with all the requirements for each routine.

In some systems other functions may also be required of the off-board communications. For example, in airbag controllers, historical data must be recorded and available for interrogation.

4.4 Limp-Home Strategies

4.4.1 Defaults

It is viewed as essential by automotive manufacturers to keep their customers mobile as far as this remains practicable. For current systems, this usually means either invoking a default state, or disabling the electronics, if it is adequately backed up by the mechanical system whose functionality is enhanced electronically. In general, smaller vehicles are probably only likely to be disabled by engine management system faults, or electrical failure. Firstly, engine management system defaults will be considered.

Typically, engine mounted sensors and connectors are regarded as vulnerable. The typical effect is that the engine fails to receive correct data, and fuelling or ignition, or both, may be compromised. The effect of this compromise could be to risk damage to either catalysts, or the engine internals such as piston crowns. It is clearly necessary for the engine management computer to invoke a default state which will allow the vehicle to limp-home, whilst only

permitting the engine to be run in a state which will avoid or minimise damage. This may involve limiting rpm; limiting ignition advance; limiting fuelling enrichment.

For simple sensor failures, it is often possible to calculate an acceptable value for the missing/ corrupted signal by making use of other related sensor information (e.g. mass air flow from rpm and throttle opening), and maintain the engine's performance sufficiently well that the driver is not aware that it is running in a default state unless he is warned that a fault exists.

A greater problem exists with systems such as automatic transmission control - especially when using a conventional dry friction plate clutch which is also controlled by the transmission ECU. A state must not be allowed to exist which risks for example trying to engage two gears at once, or trying to effect a gear change without clearing the clutch, and without correctly adjusting engine rpm. Not only would serious damage be done to the gearbox, but the vehicle would almost certainly be disabled. Indeed, if two gears were engaged at once, the vehicle would be almost certainly unmovable - the driving wheels would be locked. In the event of a fault being detected, the only safe default would be to (a) warn the driver, and (b) prevent any further gear changes being attempted. This would also, of course, have the potential of disabling the vehicle, especially if it were to approach a "Stop" sign, etc. In this event the default warning must advise the driver to stop and seek help as soon as practicable - not just allow him to proceed with abandon.

It is best for defaults to be configured so as to offer alternative sensor information or mechanical back up wherever possible, and all default definitions must be supported by a properly thought out diagnostic strategy and objectives.

4.4.2 Safety

Above all, defaults must not result in an unsafe state being allowed: the default must be to a safe state. Whilst this may appear obvious, it is not always adhered to. Take the situation where an ECU detects that a critical sensor has failed, and the default value is adopted, it is important to consider the full effects of that default on the vehicle to ensure that no additional hazards are introduced.

The same is also true of watchdog activity. Whilst the watchdog is essential, and must not be inhibited, it must not be possible to cause an action which is unsafe. Consider throttle by wire: again, during an overtaking manoeuvre, a watchdog-induced reset could result in the engine being effectively returned to idle for a period long enough to create a hazard. In the event of the watchdog being activated in the throttle by wire computer, there should be a communication route to the engine ECU which allows it to detect the situation, and at least maintain the previous engine operating conditions until equanimity is restored.

Defaults must be considered very carefully so that safety is **always** maintained whilst the vehicle is driveable, and safe-states should be derived with reference to the system hazard analysis. The default action taken must be appropriate for the controllability category of the hazards involved.

4.4.3 Hazard Analysis

It is expected that hazard analysis will be carried out on any safety-related system. It is recommended that default states are included in hazard analysis. Potentially unsafe defaults, or combinations of default states, should be inhibited or corrective action taken.

Hazard analysis of default states should consider potential driving situations, and how the default states, or combinations of default states, interact with those situations.

Hazard analysis should also consider the effects of watchdog activity, and how a watchdog initiated reset should be configured, so as to maintain a safe state.

4.4.4 Legislation

Legislation may influence limp-home or default states almost as much as safety. If a vehicle is fitted with a multiplex wiring system, it is clear that in the event of a bulb failure being detected, an alternative lamp may be substituted (if fitted). Whilst technically attractive, this is possibly "more" illegal than reporting the failed lamp to the driver, and leaving the onus on him/ her to have it fixed as soon as possible.

The same may also be true of an engine management default: it is possible that the vehicle may be driveable, but polluting either with high CO or NOx levels, or excessive smoke. In some countries this will soon be illegal (see section 6.6), and is causing vehicle manufacturers to diagnose the **onset** of illegal faults, rather than waiting for failure to occur.

Legislation may also demand that certain default states are set when a fault is detected; this may be a corollary to "Safety" above, since much legislation is about safety.

4.5 Diagnostics Tools and Development Tools

In practice, development tools are little different from off-board diagnosis tools. One engine management system, which is plug-in ROM based, substitutes a head-up display/ interrogation/ monitor unit for the ROM plug-in. This unit can simulate RAM as well as ROM, and offers snapshot modes for a wide range of parameters; modification of values in RAM; look-up table modification on-line.

Similar systems are employed for other vehicle systems, such as ABS and airbag controllers.

Development systems are rather powerful tools; if used on public roads, they must be treated with caution. The ability to change ROM values on-line creates the possibility of an unstable situation occurring, which may compromise the vehicle’s operation. Partly for this reason, some development tools require a security code to limit access for the modification of ROM data.

It is often the case that on-board diagnostics are the last thing to be added to an ECU's functionality as a production release approaches. Notwithstanding the problems this can cause as far as disciplined software development is concerned, there is also another issue which is not usually considered. It is important to detect and correct problems **during development** rather than waiting until production. During development is when they are needed most, this is where problems are expected to arise. Unless there is adequate provision for diagnosis it can be very difficult to resolve problems with complex systems. It is therefore recommended that on-board diagnostics and tools are available at the first release of software to assist the development process, and not just viewed as post production support (where of course one would hope that there are no problems).

4.5.1 Visibility of System Parameters

Development tools have visibility of most system parameters, even including keys and checksums. It may be undesirable to allow access to such data to dealer staff who merely wish to diagnose trouble. Even more so, it is undesirable for some information to be accessible to roadside repair personnel.

It also may be undesirable for dealer staff and others to have access to all trouble codes - some are included for aiding design staff in investigating warranty returned units for example.

4.5.2 Purchasing Implications

Obviously, diagnostic tools must be purchased which comply with legislative requirements where these exist (e.g. OBD II); they must also be compatible with the systems to be diagnosed. Older types of diagnosis tools are often stand alone in concept; recent tools are personalised either by CD or a ROM-based personality module. these tools are usually purchased from the organisation who supply the vehicle system.

A new generation of tools is emerging: a relatively small device incorporating an LCD screen, which accepts CD-ROMs allowing personalisation to any vehicle, providing a CD- ROM is available (at least, in theory). These tools are exciting interest from the roadside repair organisations, who have to quite a large extent been excluded from obtaining data from vehicle system manufacturers, or vehicle manufacturers, to enable at least roadside diagnosis of trouble codes. These devices may become attractive to vehicle manufacturers faced with using a variety of suppliers for systems as well as a variety of types of systems.

5. Tools and Testing

5.1 Introduction

As modern vehicles become more complex there is a need for increasingly thorough testing at all stages of development to ensure that a safe and reliable product ensues.

Whilst the aim should be to design-in quality there will always be a need to test the completed design for correct functionality. Note that the later in the design cycle that bugs are detected the more costly they tend to be. Also note that the more complex a system the more difficult it is to detect bugs, and vehicle integration is concerned with building up more complex systems from smaller sub-systems.

The idealised and pragmatic aspects of the development processes are driving manufacturers to seek powerful (computer based) support tools to assist in the design and integration of complex vehicle systems. Although such tools assist in the validation of the vehicle electronic systems the question now arises as to the validation of the tools themselves.

5.2 Interface Testing

Assume that the vehicle electronics are composed of several electronic control units (ECUs). The ECUs will need to acquire input data and from this derive a set of output demands. The input/output interface may be in the form of discrete wires (or pipes (e.g. vacuum pipes) or embedded visual indicators (e.g. lamps, LCDs)), a point to point communications link with another ECU or a vehicle wide communications system ("multiplexing"). The view the vehicle has of an ECU is largely determined by its interface. Therefore, the interface behaviour of the ECU must be tested to ensure that it meets the requirements of the vehicle. This is especially the case with regard to communications links between ECUs.

The drive of interface testing is to assure that the ECU fully complies with a published (standard) interface. Any discrepancies between the published standard and the ECU must be annotated in the test results documentation for the ECU. It should be noted that some standards are more advisory than prescriptive. In these cases the precise interpretation of the standard must be agreed between the vehicle integrator and the ECU supplier.

For standard interfaces the use of a defined test suite to qualify the ECU should be aimed for. This approach is similar in concept to the ACVC (Ada Compiler Validation Capability) tests for Ada compilers. These tests are a documented set of tasks that the Ada compiler must complete successfully. In the case of ECU interfaces it should be possible to define a set of hardware and software tasks that the ECU must perform successfully; these tests would then provide a basis for acceptance of the ECU by the vehicle integrator.

5.2.1 Speed

A prime parameter of inter-ECU links is the speed of the link. At a low level the bit rate of the ECU interface must be within limits for the bit rate of the communications bus (e.g. 1200 baud, ±10%). Above this level there is a concern that the ECU hardware and software can receive and transmit vehicle messages at a rate that meets the requirements of the integrated system under all specified conditions (environmental as well as functional). Once the low-level speed of the ECU has been checked the message timing parameters (e.g. answer-back times, maximum message delays) can be assessed.

5.2.2 Initialisation

To ensure a reliable, predictable system it is necessary to ensure that the interface characteristics are defined at every power-up or reset event (e.g. by having the processor explicitly write initialisation commands or data patterns to the interface electronics).

5.2.3 Tolerance to Errors

Any communications link must be tolerant of errors, especially where the link is related to real-time safety-related (or safety-critical) functions. Errors can manifest themselves on many levels, these levels are related to the ISO-OSI seven layer model (see section 3.5) A condensed three layer model is proposed; it is in this context that the potential errors are classified below :-

Physical Layer:

- This layer is concerned with the hardware parameters concerned with a particular communications standard (connectors, transceiver circuitry, drive levels, etc.).

- It is likely that these low-level functions will be performed by a chip-set for the desired protocol. Whether this is the case or not it must be ascertained that the hardware meets the requirements of the appropriate communications standard. The standard may identify failure cases of the hardware which must not impact on the operation of the exterior communications bus; assurance must be sought that these requirements have been met (e.g. hardware timeouts on bus usage). Not only does each ECU have a functional responsibility it has a non-functional responsibility such that a common communications bus is not dragged down by one faulty unit (such a problem not only causes the bus to fail but determination of the faulty unit could be very involved (a bit like finding out which Christmas tree light has failed)).

Transfer Layer:

- Low-level parity, frame length and protocol concerns may be implemented in hardware.

Note: It is quite common for vehicle communications interfaces to be implemented in hardware. However, on certain links (e.g. low speed) the low-level transfer layer functions could be implemented in software.

- Assurance of these functions fall into the same area as Physical Layer functions (above).

- It is likely that ECU software will be involved with higher level transfer functions (such as message retry).
- The transfer layer should be robust against Physical Layer and Application Layer errors or anomalous behaviour.

Application Layer:

- This layer is concerned with the use of data received (or the generation of data to be transmitted) by the application software within the ECU (i.e. the actual function of the ECU). Apart from verifying that the (software) interface between the Application Layer and the Transfer Layer is adhered to the content of the application layer does not concern us here.

- Compliance of the interface against the design interface may be enforced through implementation language facilities (e.g. Ada package specifications) or it may be assessed by review. Note that it is compliance against the agreed design interface, not necessarily a published standard, that must be assessed. As has already been stated, some standards allow for variability in interpretation, such variability must be defined in the agreed design interface.

The communications link (from the Physical to the Application layers) should be evaluated for correct operation before the ECU is released for integration into the vehicle system. To determine the test cases needed to characterise the link (and thus assess it against the specific published protocol standard) techniques such as Fault Tree Analysis may prove useful. For each requirement of the standard it would be possible to construct a Fault Tree describing contributions to transgressions of the requirements, from these contributions the test conditions can be generated. If a chip-set is used to implement the protocol then documentation should be obtained from the manufacturer stating which sections of the appropriate standard are complied with, and which are not met.

Each level should deal with the errors appropriate to that level, e.g. the Application Level should not deal with protocol errors. likewise the Transfer Level should not attempt to determine if the data in a valid message is in-range.

5.3 Simulation of Networks

With a network of intercommunicating tasks (ECUs) it is sometimes difficult to arrive at the optimum network architecture (with respect to network topology, link capacity, fault tolerance,

etc.). It would be advantageous to use a computer based tool to simulate proposed networks. The required functions can then be distributed over several different network architectures, simulations performed (using the same functional input data for each network) and the various architectures can be evaluated against one another.

To be useful the simulations must be repeatable, configurable and be documented. The documentation should, ideally, be generated automatically by the simulation tool.

The accuracy of the simulation must be assessed by comparing the simulation results against bench or vehicle tests. The degree of correlation will indicate the quality of the simulation (which is useful to know for future projects or if modifications are required to a previously simulated network). To make this comparison easier it is useful if the simulation input set can be used to drive the actual hardware (via suitable rigs) and the data collected from the rigs is similar to the data output by the simulation tool.

Any assumptions made during the modelling of the system must be accurately and completely documented.

5.3.1 Data Bandwidth

To assess required data link capacity it is necessary to generate representative message distributions for the ECUs on a particular link or bus. An upper limit for required data bandwidth can be calculated for a link, knowing the message traffic that the vehicle functions require but this may indicate that a link of relatively high capacity is required.

By simulating a link of lower capacity than required (to perhaps lower the cost on the vehicle) it should be possible to determine the effects such a link would have on message transport delay.

To be able to assess such changes it is necessary to be able to recreate the input stimuli to the simulation such that the results of one simulation are directly comparable with another. The input stimuli may be determined by random distribution, schedule or specific critical cases (start-up, link failures, etc.).

Experimentation may highlight the need to give certain messages priority on the bus (and thus delay less important messages), or it may indicate the need to have separate communication facilities for parameters associated with safety-critical sub-systems.

5.3.2 Errors

How the vehicle system copes with errors is of major interest, since the occurrence of some errors during the lifetime of the system is inevitable. Some errors are related to degradation of the hardware (introduction of noise leading to bit changes, link failure due to failed connectors) and some may relate to design errors in the way that ECUs manage the message protocols. Simulation can address the effect of noise or link failure but to address design/implementation errors is really beyond the scope of simulation.

Ideally it should be possible to specify error insertions as part of the input set, possibly in a script (i.e. a file of commands that can be repeatably executed by the simulator).

5.3.3 Animation

For greatest support for the designer the simulation tool should offer a (graphical) animated view of what is happening. If the designer can directly interact with this animation then so much the better. The use of an animated interface speeds development (of the animation and the target system) and the dissemination of ideas.

To achieve the greatest simulation speed it will probably be necessary to turn the animated display off and run in a batch mode. This is no great problem as the more detailed simulations will involve defined input stimuli and the animated interface will be used for what-if evaluations and prototyping activities.

5.4 Integration Tests

Each programmable component in the system must have its own specifications for both hardware and software, and hence it is possible to perform the usual lifecycle verification and testing on each component separately against these specifications. e.g. software module tests, integration tests (modules/software and software/hardware) and "system" tests (where "system" refers to the single programmable component). These should be carried out by the supplier prior to delivery of each programmable component. (See MISRA ST6-Verification and Validation).

Because of the nature of multiplexed systems an additional test phase is necessary - the integration of individual programmable components into the complete system. Before this can take place the lifecycle tests on the components must be completed, and the vehicle manufacturer must have confidence that the functionality and communications requirements of each programmable component are as specified. i.e. all the data expected out of, and actioned by a programmable component is correct. Once there is sufficient confidence in each individual component of the system, then it is possible to begin to connect them together via the communications bus. This could be done stepwise, by connecting and verifying only two components to begin with and then adding one at a time until the complete system is formed, reverifying at each step. The verification should check, as a minimum, the compatibility of communication and the i/o interfaces. It is recommended that this is done on a bench simulator rig prior to attempting installation in a prototype vehiclee.

It must remain the responsibility of the vehicle manufacturer to plan, perform and sign-off system integration tests whereby individual programmable, communicating components are linked together as a full system. Again, this is especially true if more than one supplier is involved.

5.5 Systems Tests

Ultimately the results that provide the largest contribution to confidence are those obtained from the systems running in the target vehicle. However, it is useful during development to have bench test rigs which offer more convenient access to test points and diagnostic data. In the aerospace industry it is common for large systems (even entire aircraft avionics systems) to be integrated on a highly instrumented "Iron Bird" test rig.

Such test rigs have ECUs, correct length wiring, mechanics, etc. installed in a form that is representative of the aircraft installation. The rig is used to shake-down the system, detect problems and provide an environment where highly stressful error scenarios can be played out without endangering lives or aircraft.

If bench test rigs are used it is vital to qualify their outputs against the results obtained from in-vehicle testing. This will allow the accuracy of the test environment provided by the test rigs to be assessed which has impacts for future projects and modifications to existing systems.

In addition to the normal tests carried out on the vehicle electrical system at pre-production stage, the following additional tests are suggested for integrated systems:

- If the bus signals are carried on twisted pair, short each of the pair in turn to battery and ground in succession and check that communication still works via the other wire.
  Note: Most twisted pair networks are specified to operate in this manner.

- Use a bus simulator to supply data to each node that violates its context and plausibility checking rules, and check for the expected behaviour from each module.

- Check the behaviour of the whole vehicle with disconnections in the bus at various points (particularly those points identified as having a higher risk of failure such as connectors).

5.6 Compilers and Languages

Some safety-critical software pundits deprecate the use of ’C’ due to its incomplete ISO definition resulting in many aspects of the language being undefined, unspecified or implementation specific. In these aspects it is viewed as being weaker than assembler. The advent of C++ is regarded by some with horror as the language specification is even weaker than that for ’C’ and elements of the compiler are becoming very complex.

Languages recommended for high-integrity applications are ISO Pascal subsets, Ada subsets and Modula-2 subsets. For lower integrity applications the above languages can be used with minimal restrictions and structured assembly languages are added to the list. ’C’ and

unrestricted assembly languages are strongly discouraged for applications where safety is an issue.

Note that for the higher-integrity applications the languages should be supported by validation tools (e.g static code analyzers), see [11].

In the past, the processors used in the automotive industry have tended to be small (their selection being driven by component cost). Consequently the languages available for such processors have been assembler with the possible options of PL/M or 'C'. Recent trends in vehicle electronics have seen larger processors appearing in ECUs but the languages available have remained largely the same.

It is unlikely that the processor vendors will sponsor support of Ada for small application devices, and it is probably unrealistic to expect to fit even a subset of Ada into the memory budgets of many ECUs. This is unfortunate as a good deal of work has been put into analysis of Ada for safety-related applications.

Bearing the above factors in mind it is likely that the application will be produced in 'C' (possibly PL/M) with assembler modules for time critical functions, or be produced in assembler alone. The problem is now to ensure testability and reliability of the application with these languages.

A computer language should not be selected in isolation, the set of language features you use and the tool support for that language are just as important.

It has been stated with 'C' [11] that it is difficult to define a safe subset of language features that avoid the undefined areas of the ISO specification.

Assembler offers the programmer the most intimate interface with the processor. It is this accessibility that offers the greatest versatility and the greatest risk. Approaches to limiting the risk may take the form of :-

- Banning the use of certain machine (or language) instructions.
  (Restrictions should be described in the coding standards for the project. Tool support to detect transgressions from the coding standards should be sought.)

- Enforcing the use of defined interfaces for access to data structures and i/o.

- Programming with proven macros rather than directly in the native assembler.
  (this offers the possibility of porting applications relatively easily between different targets but will result in more target code than hand crafted assembler).

- Programming with proven libraries. This is an extension of the macro approach described above, and may even be combined with it.

Using macro or object libraries raises some concerns. To confidently use libraries strong change control and status accounting is required. For each issue/version of a library it should be possible to discover :-

- The issue/version of the library components.
- The testing/approval status of the library components
  (traceable back to particular test schedules, results and tools used to assess the component). This is particularly relevant as old components may not have been tested to the same degree as new components.
- The issue/version of tools used to build library components
  (e.g. the version of an assembler used to build an object library component).
- The issue/version of the librarian tool used to create and modify the library.
- The update history of the library.

Rather than trying to restrict existing languages to make them safe a more radical approach is to define and create a language appropriate to a high-integrity environment. This is an extension of the use of macros suggested above. It should be realised that the investment required to define, create and maintain a language (and associated support tools is high).

Tool support can determine the ultimate utility of a particular language. It is better to have a less than ideal language supported by tools for debugging, testing and documentation than an ideal language with little or no support. Likewise it is better to have well trained, technically aware engineers who understand how to safely apply a particular language than have only some understanding of a new language.

To assist the day-to-day implementation of software a site (or project) specific Programmer's Manual or coding standard should be produced detailing how a particular language is to be applied. Dangerous language constructs (e.g. unchecked CASE select expressions in PL/M-96) should be listed - and acceptable workarounds described. To meet TickIT and BS5750 this type of operational manual is required. To assist detection of banned constructs the use of tools should be considered. (see section 5.7.2).

Perhaps a more relevant attribute for an application is not what language it is produced in but how testable it is. The application should be designed and implemented in a modular fashion. This approach permits the testing and integration activities to be planned in advance. If it proves difficult to write a test plan for the design then this could be an indication that the structure of the design needs changing. Having a simple control flow within each module makes verification of function simpler. The use of a cyclomatic complexity metric can be useful here to indicate when a module is turning into spaghetti code. There is a tendency for some programmers to produce code with speed or memory usage as a prime driver. Unless

there is a proven need for time and/or memory saving in a particular module then it should be coded with clarity and testability as the paramount concerns (this will also assist in maintainability).

With some languages it may be necessary to make program variables global rather than local to a particular function so that they are visible to their associated test tools. Such requirements should be stated in the project coding standards.

5.7 Tools for Consideration

The following tools are those which may be considered as part of the process for developing integrated systems with multiplexed busses :

- Bus simulators*
  To be able to simulate the traffic on a communications network to evaluate its performance with respect to bus loading, speed, noise, message latencies and priorities.

- Bus analyzers/emulators*
  To be able to transmit/receive messages to/from the network, monitor the error performance, log data in real-time and provide statistic on message scheduling.

- Schedule driven test support
  To be able to carry out regression testing it is necessary to be able to repeat test stimuli. To reduce the cost of repeat testing it is a considerable advantage to have the capability to run a complete sequence of tests unattended.

- Coding standards checker
  A tool to check that the coder has not employed constructions prohibited by the coding standards relieves the need for laborious manual checking.

- Cross reference tool
  Source code cross reference tools, particularly where they indicate reads and writes, are useful in checking that the source code does not access data in a different way to the design.

- Complexity metric tool
  During the coding phase it is useful to assess the complexity (no. of nodes, branches, etc.) of the source code.

- Static code analyzers
  eg. SPADE, MALPAS

- Dynamic code analyzers

- Test coverage analysis

  With a high-level language this will probably be restricted to coverage analysis of the high-level source code, rather than the generated object code. Test coverage usually requires the source code to be instrumented before being built.

- Functional checking

  Source code can have pre/post conditions embedded in it (usually as comments). During dynamic testing the assertions in these special comments can be checked against the actual performance of the code.

- Timing analysis tool

  The maximum execution time of functions in a real-time system should be defined in the design documentation. The (worst case) performance of the functions must be measured on the target hardware and compared with the design requirements. With some languages (such as assembler) it is possible to make an assessment of the worst case execution time by analysis rather than measurement. However, in these days of pipelined and cached processors these assessments may not represent the actual execution time in hardware.

(* Bus simulators/emulators/analyzers for CAN, VAN, J1850 and ISO 9141 are available from I+ME. The UK agents are Nohau Ltd., telephone (01962) 733140, and emulators/analyzers for CAN from Softing GmbH in Germany, phone 00 49 89-41 30040).

5.7.1 CASE Tool Applications

In the analysis/design arena there are a couple of competing methodologies (the Structured approach and the newer Object Oriented Analysis (OOA) approach). The newer paradigm always tends to get the most comment in the technical press, however that should not mean that a user of an existing methodology should switch to the latest fashion straight away (if at all).

Note: It is possible to use OOA to analyze a set of requirements. This does not mean that Object Oriented Design (OOD) and an object oriented implementation language such as C++ has to be used.

If a group is using a particular methodology (such as RTSASD (Real Time Structured Analysis Structured Design)) it is likely that they have made a substantial investment. This investment takes several forms :-

- Cost of tools

- Cost of training
- In-house expertise
- How to apply the methodology to the products
- How to get the best out of the CASE tool (both in knowledge of the tool's capabilities and any locally produced utilities (perhaps for data interchange with other tools))

While a new methodology may be perceived to offer benefits, the cost of achieving the level of process integration acquired with the old methodology and associated toolset should be recognised. Perhaps the ideal way to evaluate a new methodology/tool is to acquire the CASE tool (on a temporary basis) and run the new methodology in parallel with the existing methodology on a REAL project. This will not only show how the methodology works but also how well the tool supports it and how robust and reliable the tool is. The other crucial concern is how easily automated transfer of data in and out of the CASE tool to other tools can be implemented (including documentation tools such as Interleaf or Framemaker).

Before the expenditure of large sums of money on a CASE tool (some cost £10,000s/seat) it is wise to evaluate the supplier of the tool. CASE tools are complex software systems, if the supplier has to contact the producer of the tool for every reported problem this can substantially slow down the resolution of difficulties. Therefore as well as evaluating the functionality of the CASE tool also evaluate the supplier for :-

- Cost of support
- Availability of support (is the support desk in the USA with all the problems of time difference?)
- Quality of support
- When you raise a problem with the support desk are you given a problem log number?
- How long does it take to resolve problems?
- Is the supplied documentation adequate?
- Stability of supplier
- Is the user base large?
- Is there a product user group?

- Do the suppliers/producers work to a defined quality system? (e.g. do they have BS5750 or ISO9000 approval).

- Which development platforms is the tool available for? (some tools are better supported on UNIX than VMS; UNIX is perceived to be more secure than VMS; PC applications tend to be cheaper)

- Does the choice of a tool mandate the purchase of a new development platform?

- If the tool is available for multiple platforms are the work files portable between these platforms? (this can simplify multi-team working)

One of the recognised problems with systems development is that of getting the requirements right in the first place. There are many tools for assisting with structured analysis, coding, debugging, etc. with little emphasis on the front end' of the process. Some thought should be given to requirements animation. Animation of a problem definition can lead to clarification or development of the requirements before the analysis, design and implementation phases start. Requirements animation may be possible using some of the tools employed for network simulation (see section 5.3.3).

5.7.2 Test Analysis Support

CASE tools can offer considerable support for test generation and static and dynamic code analysis.

Note: The features described are abstracted from the documentation of the TESTBED tool (developed by Liverpool Data Research Associates and marketed by Program Analyzers Ltd.). Description of these features does not imply that TESTBED is the only available tool or that it is in some way implicitly recommended by MISRA.

Sophisticated test support tools can do a lot more than just run tests on the target. Static analysis is carried out on the source code to determine :-

- Complexity metrics
- Data Flow anomalies (e.g. variable read before write)
- Information Flow (detect variable dependencies and validate annotated dependencies)
- Cross Reference (showing call tree and variable usage)
- Compliance to language sub-set definition.

Note that the host language may be restricted to a sub-set that can be analyzed.

All of this information is valuable during the coding phase to assist the programmer in locating and eliminating problems before the code is dynamically tested.

When the code is ready for dynamic testing it must be loaded into a (representative) target system. The test CASE tool runs on a host computer system that communicates with the target system via a serial link. The code loaded into the target is built from source that has been instrumented to provide the test analysis tools with information on the paths taken, etc. The host system then causes test schedules to be run against the target code and collects the results. From these results test effectiveness ratios for statement coverage, branch coverage and Linear Code Sequence And Jump (LCSAJ) coverage can be calculated.

Dynamic conformance with predefined assertions (pre-and post-conditions) can also be evaluated. The assertions take the form of special comments embedded in the source code, the truth of these is evaluated during dynamic testing.

Administrative functions are available to store the static and dynamic analyses for comparison with other test runs and the production of textual and graphical reports.

From the above it can be seen that certain facilities must be available for the tool to operate (in the target) :-

- The target must have a serial link available for the use of test equipment. If this link is shared with some other function then obviously that function cannot be dynamically tested with the standard test tool.

- The target must have enough memory to accommodate the overhead of instrumented code as well as the test kernel handling serial link communications and code invocation in response to host system commands.

To avoid impacting the ECU in this way it would be preferable to replace the serial link and object code instrumentation with an emulator based tool. The target software would then be unchanged for the test runs and would execute in real time (the overhead due to host link communications and data gathering would be eliminated). Once data is collected in this way it then becomes possible to collect execution time profiles for the target software, watch stack usage, etc. These parameters can then be compared with the memory and timing budgets in the design documentation. This type of facility would require that the host based analysis tools interface with a (specific) emulator.

It is a productivity enhancement if the tools selected can be run in a self-hosted manner as well as host-target. This allows the code implementors to statically analyze their code and the testing team to develop their test schedules prior to target hardware becoming available.

Other tools exist to assist in the definition of the test data required to fully exercise the source code under consideration. As with most of these tools the important thing is to have an easy transfer of data between utilities. Despite effort in this area it still seems to be an area in need of attention.

5.8 The Management of Integrated Systems Testing

The potential complexity of multiplexed vehicle systems requires that the implementation of such systems be carefully planned.

Integration of the vehicle systems should be planned in a phased manner. Confidence in a sub-set of the overall system must be gained before adding extra ECUs. This may require test equipment to simulate operation of missing ECUs (or the architectural and individual ECU designs must accommodate partial installations).

For each phase of the integration process a test plan should be developed detailing the tests necessary to validate correct operation of the (partial) system. The test plans must detail the test equipment and other facilities required to carry out the tests. It is to be hoped that test equipment developed for network integration will be applicable across many projects (and thus reduce the cost burden on any particular project). Note that the test equipment required for vehicle network integration could be quite sophisticated, requiring facilities for ECU internal operation monitoring, network monitoring, fault insertion, bus usage, message logging, etc. (See section 3.5.7, 3.6).

Note that correct functionality of test equipment is at least as important as ECU functionality, therefore software and hardware in test equipment must follow a similar process lifecycle to the ECUs themselves (including reviews, configuration control, etc.).

To assist in proving the operation of the network early on it may be feasible to test the network interface elements of the ECUs prior to the prime functionality of the ECUs being implemented. Thus integration of the networking elements can be established without the extra complication of the ECU functionality confusing the picture.

It is likely that the vehicle manufacturer will have subcontracted ECU development. Under these circumstances it is vital that a defined group owns the network and the associated integration task. It is probably best that the vehicle manufacturer takes on this vehicle level task. However, if the manufacturer does not have the expertise to carry out this job then it must still ensure that the subcontractor nominated to own the task is effectively communicating with the other ECU subcontractors. This is a potential problem area as the subcontractors could be competitors. The vehicle manufacturer will need to be completely aware of all the technical issues, e.g. data gathered during the integration phase (such as network loading) must be reported back to the design authority (e.g. the vehicle manufacturer) so that it can be compared against predictions arising from the system analysis.

The vehicle manufacturer must also monitor progress towards resolving any problems and meeting integration milestones, thus while it may be able to delegate ownership of the (technical) integration task it cannot delegate overall management responsibility.

6. Off-Board Diagnostics

6.1 Introduction

As with any item, the increasingly complex vehicle will over time be subject to failures. Whilst the vehicle design will try to minimise the effect of these failures, they must still be recovered from, as and when they occur - even if the fault is simply the vehicle running out of fuel.

When the fault appears to be a little more obscure, the technician is faced with a complex and expensive vehicle to progressively disassemble until the fault is identified. Clearly this is undesirable and advice has to be given as to the correct way to deal with this situation. With the aid of workshop manuals and simple test equipment, a skilled technician should be able to get the vehicle back on the road - even if it was unclear as to what the fault actually was. This can be seen with warranty return items which have to be classified as "No Fault Found".

When the fault is not traceable, due to complexity or functionality (e.g. ABS not working correctly) a far higher quality of tool is required to test and interrogate the Electronic Control Units. After the vehicle has been uniquely identified, it is possible to communicate with individual (or in some cases with a group of) ECU’s to narrow down the item that has caused the failure. With the aid of custom instructions for the particular vehicle, (rather than expecting the technician to cross-reference several disparate pages), a quicker and more robust service can be expected.

With the growth in non-franchised dealers adding to the existing pressure from the road side repairers, any interface to off-board diagnostics should be via a widely agreed standard. For example, an ISO 9141 interface to a single ECU could become the gateway into the other ECU’s on the vehicle. This would allow the use of generic "Scan Tools" which, when added together with relevant proprietary information, could become powerful diagnostics aides.

In those cases where the fault is intermittent and not traceable via any of the service bay based tools, a signal monitor and recorder (also known as a flight recorder - see section 6.5.1) could be used to directly record the activity. This is a highly skilled task, both in setting up the equipment and more importantly in analysing the data - after all there is little point in recording information that will be intentionally ignored.

The same technology used to service the vehicle could also be used to build it in the first place. Ensuring that the design is aimed at making the manufacturing process easier, such as using programmable ECUs on the production line, yields improved productivity and quality in a reduced time.

Diagnostics should be designed in as part of the system requirements and not added in later. This ensures the diagnostic requirements can be met at the initial design stages and any changes required to support diagnostics can be properly assessed. Designing diagnostics in at a later stage can increase the overall resource required to implement the required

functionality, and lead to the possibility of reduced diagnostics capabilities, poor overall quality and a potential loss of system integrity.

6.2 Service Diagnostics Systems

Generally, on-board diagnostics are limited due to processing and memory requirements. The ability of on-board diagnostics to diagnose faults in mechanical systems of the vehicle, or indirectly controlled electrical systems is generally limited.

Off-board service diagnostic systems however have the ability to not only retrieve data from on-board systems and apply on-board tests, but also to diagnose faults using additional systems to measure the parameters outside the domain of the vehicle control units.

Intermittent fault detection and high speed real time data analysis are difficult to achieve via service diagnostic systems. The aim of service diagnostics in these cases should be information retrieval from the on-board diagnostics system. This also infers that any diagnostics requirements to achieve the above are designed in as part of the system requirements.

Service bay tools could also be used to perform special tests which involve the downloading of test routines and/or data to on-board systems. The integrity implications of this concept are brought into question for safety critical systems. If the downloaded test routine can affect the normal operation of the system, e.g. change the operating parameters of a chassis control system, then care must be taken in ensuring the unit is configured back to normal operation before ending the diagnostic session. This can be effected by the downloaded routines not overwriting the normal program, and the routine only executing in the current diagnostic session. Exiting the diagnostic session, either by controlled exit or power down, must return the unit to normal operation. e.g. by turning the ignition off /on.

The application of off-board diagnostics tools must be supported by adequately trained operators and sensible application by vehicle manufacturers of such tools to franchised dealers.

The relative importance of service diagnostic systems can be over estimated, especially when they become an interesting technical challenge, and they can be a large sink on resources and funds for their development and maintenance. In fact the resources and funds may have been better employed in improving the original design of the vehicle to improve quality and reliability to reduce the need for complex off-board diagnostics in the first place.

6.3 Repair Instructions and Fault Finding

Service diagnostics are required to cater for both expert and inexperienced technicians. For the inexperienced technician it is appropriate to do a full system test, and when a fault is

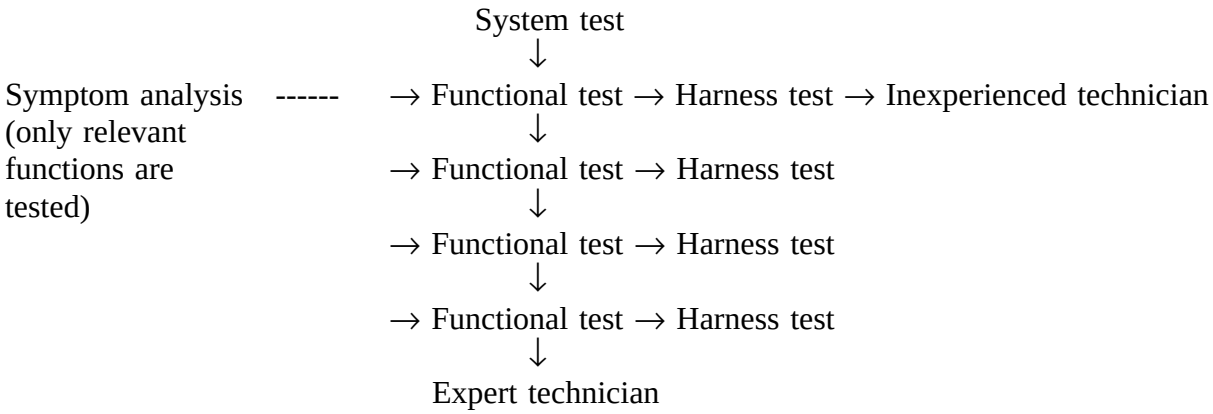
found on a vehicle to lead him down a fault tree until the failed component (lowest replaceable unit) is identified, e.g. sensor failed, open circuit wire, etc.

The expert technician will generally only require the fault to be identified to a functional block within the system. With electronic control units the functional block containing the fault can often be identified from fault codes. It would be inappropriate to force an expert technician down a fault tree, and the diagnostic tools available should allow identification of a component failure, e.g. the ability to read sensor values, drive ECU outputs, take voltage measurements, etc.

The type of diagnostics implemented will also be system dependent. For example, with an engine management system the best approach may be a symptom driven diagnostic, such as "poor running when cold".

With body systems individual functions can often be tested, e.g. left hand front window non operational.

Signature analysis for harnesses can be derived from design FMEAs and the diagnostics implemented from this. Care must be taken in doing this to ensure that the implementation is tied to the harness in such a way so that a minor harness change does not require a complete rewrite of the off-board diagnostics software.



(Notes : A system test is made up of a suite of functional tests. Symptom analysis selects only the required functional tests for a particular fault type. Experienced technicians will tend only to perform functional tests, and then use their knowledge of the vehicle to find the fault. Inexperienced technicians, however, may need guidance down to individual circuit and wire detail.)

6.3.1 Diagnostics of a Communications (Multiplex) Network

As well as using the communications network as a medium for performing diagnostics, it is also necessary to consider what to do when the communications network itself fails.

Assuming that the network is designed to work, there are only 2 possible main failure modes - loss of communication and corrupt communication.

Loss of communication can be due to wiring or connector faults on the bus, failure of the transceiver circuit, power loss or complete failure of an individual control unit. Corrupt communication can occur if a control unit logically malfunctions or by a harness/connector problem that degrades the bus performance (e.g. loss of one wire of a twisted pair).

Steps must be taken to ensure that a technician can approach this problem correctly. The traditional "test bulb" techniques are no longer valid for communication busses. It is recommended that a hierarchy of diagnostic routines are considered. An example of such a suite of routines is :

- Physical check of voltage / current levels expected on the bus to determine possible harness faults. It may be necessary to disconnect all control units from the bus to finally trace the fault. These are the most difficult faults to locate, but are easily fixed.

- Check of the message identifiers expected from each control unit. One valid message from each control unit will confirm that the bus is working; this can be used to locate missing control units e.g. due to power failure.

- The ability to test each control unit individually e.g. send it data that it would normally receive from the bus.

- The ability to emulate each control unit individually e.g. remove the control unit and replace it with the test tool to transmit the data normally transmitted by the control unit.

- Access and display of individual data on the bus.

Together with on-board network monitoring (e.g. low priority token messages and timeout checks) and the provision of careful guidance to a technician it should be possible to quickly and accurately locate faults that exist on multiplex networks.

6.3.2 Roadside Repair and Non-franchised Repairers

Access to information for non-franchised repairers and for roadside repair throughout the life of the vehicle is required. Without this information a vehicle may be perceived by customers as a product which is not easy to repair when it goes wrong. Information about mechanical repair procedures can be deduced from the vehicle itself, but off-board diagnostics of the vehicle’s systems will require proprietary information about the vehicle and/or equipment to be made available by the manufacturer. This could be construed as compromising the franchised dealers position in the repair market for that vehicle. Legislation, as in the US, can be brought to bear to force manufacturers to support a common diagnostics protocol e.g. SAE J1978 OBD II Scan tool for emission related items.

Access to the technical information required to diagnose systems other than these is seen to be proprietary to the manufacturer and requires support in terms of training and manufacturers' backup services. Due to the wide range and variety of systems employed across manufacturers' vehicles, it is not seen at present that roadside repair or non-franchised repairers can gain expertise equivalent to that of a limited number of franchised dealers who have been properly trained. In view of this, vehicle manufacturers should warn their customers that it is important to ensure that safety-related systems are only attended to by authorised and adequately trained repairers.

6.4 Relationship to Warranty

6.4.1 Cost of Repair

The effectiveness of diagnostics can be measured via the warranty claims returning to a vehicle manufacturer. Warranty costs can be large (especially with the advent of buy-back, or "lemon", law in the USA) and diagnostics are seen as a route to reducing these costs. Although a reasonable assumption to make, it is not entirely obvious how this works and it is not totally true.

The warranty cost is basically affected in four ways by improved diagnostics.

- Reduction in the number of control units that are replaced under warranty and classed as no fault found, i.e. a control unit was functioning correctly, but another fault somewhere in the system caused the repairer to suspect it. More accurate and detailed diagnostics should help pinpoint the item that has actually failed.

- Faster diagnosis of the problem is often possible with electronic tools. The time for a repair is also often charged to a vehicle manufacturer for a warranty claim.

- More diagnostics can lead to more, previously dormant, failures being detected. Hence it must be remembered that in some circumstances better diagnostics can actually lead to an **increased** warranty bill.

- Prevention of damage to other component by a persistent fault, e.g. catalyst damage due to an engine management fault.

Overall the recommendation must be that it is better to engineer reliability and robustness in to the design, and overall it may be cheaper to do this than accept the warranty costs that would result.

(It is possible to take an Noise/Vibration/Harshness signature of each vehicle leaving the production line. This could then be used to assess the vehicle objectively in the event of a complaint relating to engine/drivetrain noise, and thus eliminate unnecessary warranty work.)

6.4.2 Lowest Replaceable Unit

The objective of an off-board diagnostics tool is to quickly and accurately locate a problem and to direct the technician to the component which must be changed to effect a repair. This brings the concept of lowest replaceable unit to the fore.

A lowest replaceable unit is usually a component that carries a unique vehicle manufacturers part number and can be ordered as an item by the technician. Often complete assemblies are the lowest replaceable unit, e.g. an ECU, because the vehicle manufacturer regards the complexity as too great to allow an individual component of an assembly to be serviceable in a reasonable time.

A manufacturer will only pay warranty on lowest replaceable units, and any tampering at a level below this (even if it is possible) will invalidate the warranty claim. Diagnostics tools therefore should carry within them a database of components which, if implicated, represent the lowest replaceable unit. The tool can also be used to remind the user not to attempt to go inside a lowest replaceable unit, particularly for safety-related and critical systems, e.g. air bag harnesses must be replaced as a whole and not repaired.

6.4.3 Analysis

Diagnostics results can be used to detect common field problems and therefore produce an entry into the warranty database. It can also be used to detect a manufacturing fault, e.g. bad batch of components, and allow either a service fix or a recall action together with a manufacturing corrective action.

The development of any off-board diagnostics tools should benefit from detailed analysis of warranty data to ensure that the type of problems encountered in the field are well understood. It is important that good information on warranty claims is available at all levels in both the vehicle manufacturers and the component suppliers, providing a feedback mechanism for design decisions.

6.4.4 Fault Ownership

Fault ownership for an isolated component failure is often not an issue. If the failure of a component leads to failure of other components, or the complexity is such that a number of components may be contributing to a failure, then ownership of the problem is more difficult, e.g. a regulator pack on an alternator failing and causing failure of a number of vehicle electrical components. The prime cause of failure in the above is obviously the alternator, therefore the liability for the warranty costs of replacing the additional failed components should be the alternator manufacturer’s. However, proving the prime cause would be difficult. In the case of control units an internal component failure should still be viewed as a warranty issue with the control unit manufacturer, who can then make the decision to recover costs for the failed component.

If the component failure is due to a software issue, e.g. the software driving a component into a failure mode, the cause is generally difficult to both prove and replicate. In practical terms the warranty costs would end up being borne by the vehicle manufacturer. Hence software must be taken as a serious contributor to warranty performance by the vehicle manufacturer and not dismissed as a supplier's problem. (See MISRA ST7-Subcontracting).

6.5 Diagnostics Tools, ISO 9141 and Multiplexing

Off-Board diagnostics tools are now available (often hand-held) which plug in to the vehicle and communicate with it using a serial interface. The large number of individual systems presently used in high specification vehicles mandate that the tool must have a means of catering for a variety of fault codes and diagnostic data. This is often currently catered for by personality modules for the tool, which may be vulnerable to damage in the field. The tool may be able to access measurement data within the system it is communicating with, and also be able to change values in RAM. Some tools have, in addition, a snapshot facility. They do, however, require a level of skill, to be used correctly.

Initially, the serial interface to tools was unique. There is however, an international standard-ISO 9141 - which lays down the physical layer of the communications protocol for diagnostic off-board tools. This type of approach offers the basis for universal diagnostic tools produced by third parties, and catering for a number of suppliers and vehicle manufacturers or models in one unit.

Recent production vehicles in Europe have adopted the use of a simple communications interface conforming to ISO 9141 to perform vehicle systems diagnostics. This was the first real application of interactive communications between off-board and on-board systems.

ISO 9141 is a standard for the physical parameters for the communications interface. All the message definitions, wake up procedures and logic needed to be defined by the vehicle manufacturer. As it is usual for different suppliers to be used for different vehicle systems there is a tendency for different application of ISO 9141 for each system, i.e. each system is standalone. Note that a single multiway connector can be provided as a means of physical access to a number of standalone control units.

Standalone diagnostics between a diagnostics tool and a control unit avoids contention arising through other control units trying to communicate at the same time. As well as being the early ISO 9141 situation, it may be the only method of supporting different protocols or physical layers between control units.

It is possible to link together each of the systems on a ISO 9141 bus, however problems can arise with wakeup modes and fast wakeup where the ISO standard is not fully supported and contention arises. Also, control units on the bus repeatedly transmitting large amounts of data, or ones which will not stop transmitting, can effectively lock the bus to communication from any other control units.

On bussed ISO 9141 systems cases have occurred where a command sequence to one control unit coincides with the fast wakeup of another control unit resulting in contention on the bus. This can be avoided by having a system specification covering the bus which all units on the bus must comply with. ISO 9141-CARB does define this for the US market for OBD II.

Bussed diagnostics provide a single point for access by a test tool. This reduces the system complexity and cost through a reduction in connectors and wires. If the protocol for communication allows a single tester to communicate to multiple control units on the bus, this potentially allows for a more system orientated approach to the diagnostics. However, for bussed systems to work, a common protocol and common message definitions must apply to all systems connected to the bus.

Vehicles may contain partial or global multiplexing of control systems (see section 3.2), using a standard communication scheme such as SAE J1850, VAN, or CAN. These standards encompass communications protocols in both hardware and software terms and they can be extended for diagnostics purposes for direct communication with off-board tools (as J1850 is for OBD II). In such situations ISO 9141 would not be required.

There may be more than one communication bus employed in the vehicle, depending on speed and complexity requirements, for example, there could be a SAE J1850 bus, and a CAN bus, particularly if a body computer is employed. Not only does the multiplexing of systems offer greater scope for diagnostic tools, it also offers greater scope for default action, and more comprehensive diagnosis in the vehicle whilst it is in service. It also, however, offers greater scope for difficulty in diagnosing faults, without the presence of comprehensive test tools.

6.5.1 Event ("Flight") Recorders

Event or "flight" recorder is the term given to a unit that is fitted to a vehicle to store dynamic vehicle parameters. The storage can be continuous or triggered from an event such as manually by the driver, or when a parameter goes out of range. At present, these devices are mostly used to determine intermittent failure modes of the vehicle systems that cannot be derived from static testing, and as such event recorders are not usually a standard vehicle fitment but are fitted by a dealer for a short period of time until the fault is located. However, another possible use for them is to provide information following an accident or traffic violation. One manufacturer is already known to be integrating them into the design of production vehicles. There are legal questions associated with built-in flight recorders, especially for their use in providing information that could be used to bring a prosecution. This subject has been extensively addressed in the EC DRIVE I project (see [12-13]).

6.6 OBD I/II, CARB, EPA and SAE Standardisation.

OBD-I was set by CARB and first issued in 1985 for the 1988 model year. This standard required the vehicle’s engine management computer to monitor the oxygen sensor, the exhaust gas recirculation (EGR) valve and the evaporative purge solenoid for proper operation and

to provide a warning to the driver if malfunction occurred. Due to the limited control effected by OBD-I, CARB issued a further set of regulations, OBD-II, scheduled to be phased in from the 1994 model year with full compliance by the 1996 model year.

From 1994, the California Air Resources Board (CARB) requires that vehicles sold in California meet the requirements of OBD II - On-Board Diagnostics II - which is embodied in a piece of legislation "California Code of Regulations Title 13, 1968.1" [14]. In order to satisfy these requirements motor manufacturers recommended to CARB that the SAE handled the standardisation. The United States Clean Air Act of November 1990 required the Environmental Protection Agency (EPA) to make regulations requiring manufacturers to install diagnostic systems on new passenger cars from 1994 model year and later model year light duty commercial vehicles. EPA published these rules in September 1991 for application to the 1994 model year. For certification, EPA will accept compliance with CARB OBD-II systems up to the 1998 model year to enable manufacturers to design one system for the market. This calls for:

- monitoring specific data for evidence of deterioration caused by wear and contamination, warning the driver when any of the aspects monitored falls outside prescribed limits

- tamper proofing of engine management systems to prevent "chipping"

- use of a standard communications protocol to allow roadside interrogation of the system

OBD II diagnostic systems monitor:

- catalyst efficiency
- engine misfires
- evaporative emissions
- exhaust gas recirculation
- exhaust gas oxygen sensors
- secondary air injection
- closed loop fuelling

A major part of the OBD-II standardisation effort by the SAE has been to define the requirements for an off-board diagnostics tool, or "scan", tool and the method by which this communicates with the vehicle. The standards cover the functionality of the tool and its interfaces with the vehicle and the communications messages which are required for both basic and extended diagnostics sessions when dealing with emissions related problem.

The UK proposal is based on the EPA requirements for the 1994 and later model year and deals with on-board diagnostic (OBD) systems for spark ignition and compression-ignition vehicles covered by the scope of directive 70/220/EEC. This proposal makes amendments to the text of Directive 70/220/EEC as last amended by Directive 91/441/EEC.

6.6.1 Relevant North American Standards

Standard (Mostly SAE)	Document Description
CARB 1968.1 (OBD II)/ EPA NPRM (OBD)	DIAGNOSTIC DOCUMENT
SAE J1850 OR ISO 9141-2	COMMUNICATIONS NETWORK INTERFACE
SAE J1930	WORDING / TERMINOLOGY
SAE J1962	DIAGNOSTIC CONNECTOR
SAE J1978	GENERIC SCAN TOOL
SAE J1979	LEGAL MESSAGE FORMAT
SAE J2008	SERVICE MANUAL FORMAT (ELECTRONIC)
SAE J2012	DIAGNOSTIC TROUBLE CODES
SAE J2186	DATA LINK SECURITY
SAE J2190	MANUFACTURER MESSAGE FORMAT
SAE J2201	UNIVERSAL INTERFACE FOR THE OBD II SCAN TOOL
SAE J2205	EXPANDED PROTOCOL FOR THE OBD II SCAN TOOL
SAE J2216	EXPLANATION OF J2008 TERMS

6.6.2 Summaries of SAE Documents

J1850 / ISO 9141-CARB

In order for the diagnostic information to be passed from the vehicle to the scan tool, several common protocols have been defined. These are ISO 9141-CARB (used by most independent manufacturer's), J1850 type 1 (GM's DLCS) and J1850 type 2 (Ford's SCP). All the protocols are class B (see section 3.1).

J1930

J1930 defines the official terminology to be used in all documents entering California for vehicle related purposes. This affects: Certification information, service handbooks, service repair manuals, etc.

Typical terminology is : lambda sensors should be called oxygen sensors, electronic control units called control modules, etc.

J1962

The CARB OBDII document (1968.1) states that all manufacturers' diagnostic systems must have the ability to communicate the same information in the same manner to the same scan tool, and that communication must take place through the same connector fitted in the same position on all vehicles.

The J1962 connector specification specifies the dimensional constraints of the connector and the pin out termination and defines the minimum durability requirements as being 200 cycles for the vehicle end and 25000 cycles for the scan tool end. The document also requires that the connector is located between the centre line of the steering wheel and the centre line of the vehicle, out of direct sight of the driver but visible to a crouched technician.

J1978

One section of the OBD II requirement is for the diagnostic information to be made available via a common scan tool. J1978 defines the operational requirements of the OBD II scan tool. CARB have dictated that if a vehicle manufacturer's diagnostic systems fail to work when tested using a J1978 type interface, they may be liable for a recall. J1978 also details the minimum functions the scan tool has to support, e.g. automatic communication interface selection, displaying of component status and fault codes, etc.

J1979 and J2190

As common scan tools and protocols have been developed it has been necessary to identify a message structure capable of accurately defining the emissions information whilst allowing manufacturers the flexibility to perform their own diagnostic routines. This has been achieved by confining necessary emissions messages to J1979 whilst more expansive messages are within J2190.

In practical terms, an independent service technician looking to repair an emission related electrical failure would use J1979 support scan tool to identify the area of failure and then use a service manual to locate and repair the failure.

An official franchised technician looking at the same problem would use a scan tool which not only supported J1979 but also J2190. This would give the technician the benefit of not only identifying the area of the fault but also locating the exact cause of failure without leaving the vehicle.

(The equivalent European standard is Key Word Protocol 2000, KWP 2000)

J2008

The US Government have proposed that from 1994 on, all written service manuals should be made available for sale to both franchised dealerships and independent service outlets simultaneously. Starting in 1996 this is to be in electronic format. J2008 is the recommended structure for service manuals stored on electronic media.

J2012

J2012 defines a method of coding faults such that the affected system, area component and core component(s) can be identified.

All codes are structured by starting with a single alpha followed by four numerics. In the case of powertrain related components the single alpha is the letter P. Body components being B, Chassis being C, etc. The first numeric following the alpha defines whether the code is an SAE fixed code, indicated by a 0, a manufacture discretionary code, indicated by a 1, or reserved codes indicated by a 2 or 3.

The following three numerics define the type of failure e.g. codes beginning with one hundred or two hundred are fuel and air meter faults; codes beginning with three hundred are ignition system or misfire faults; codes beginning with four hundred are auxiliary emission controls; codes beginning with five hundred are vehicle speed and idle control; codes beginning with six hundred are computer and output circuits; and finally codes beginning with seven hundred are transmission faults.

The document also breaks down the failures within these topics but this break down is too numerous to mention.

J2186

Many manufacturers are developing technology and facilities for programming ECUs on the production line ("on-track" programming), e.g. flash EPROM. Coupled with this is the ever present risk of unauthorised tampering and reprogramming to override diagnostic systems.

J2186 is SAE’s attempt at defining tamper detection methods such that when access is gained illegally it is recognised and tracked. This gives safeguards to manufacturers if the need arises to defend why one of their systems has failed an emission/diagnostic test.

J2201 and J2205

Because of the development of electronic communication technology and the fact that the OBDII document lives until the year 2002, the SAE have been proactive in ensuring that the standards for hardware and software are capable of surviving for that time.

J2201 and J2205 documents define communication interfaces such as bar code readers for transmission of messages, and more flexible structuring of message formats to support future needs. Although a basic tool need not have these facilities operational, each must have the capability to utilise such advancements.

6.7 System Security Issues

6.7.1 Anti-tamper checks

OBDII places a requirement on the manufacturer for the tamper prevention and detection of emission related components including the engine and transmission control units. This breaks down into two areas - physical and logical.

Physical access can be limited in a number of commonly implemented ways, for example the use of specialist screws on the control units case, seals on the case and conformal coating on the circuit board to reduce the possibility of rework.

Logical access can be implemented via the application of checksums on code, employment of masked and non-socketed memory devices. Note that checksums are relatively easy to defeat, and the sophistication of "chippers" is such that they can routinely revise checksums to suit.

Both of these methods would help prevent logical changes to the code or parameters. In vehicle immobilisation systems if multiple control units are required to communicate on a vehicle, then having the control units store an identifier unique to the others would prevent swapping a control unit to mobilise a vehicle without the other control units first learning the new identifier. The learning of new identifiers could be prevented by the use of "secure" diagnostic codes, e.g. ones which constantly change "randomly" according to a hidden algorithm.

The United States Environmental Protection Agency takes anti-tampering further than the simple "increased engine performance chips" i.e. fitment of any nonoriginal equipment or systems which could adversely affect emissions. If an additional aftermarket system has been added, e.g. air-conditioning, then the use of vehicle content identification could detect this. Items such as high-capacity alternators could be detected by alternator load sense strategies. In general the detection of nonoriginal equipment via software in the vehicle’s control systems is severely limited and the use of service database information from the manufacturer and vehicle electronic identification is the only reliable method, i.e. off- board detection.

The SAE J1724 task force suggest that a permanently attached on-board electronic memory "chip" may be the most feasible method to capture and output the specific build content of a vehicle. It would be secured to the vehicle in a protected location and read via the standard diagnostic bus/connector. A device separate from the vehicles functional computer(s) would avoid the repair problems of transferring ID to another device in the event of its replacement due to failure or upgrade. As an electronic device, subject to failure, any replacement device would have to be programmed either by visual inspection or from the service/manufacturers history. A "smart device" offers the possibility of recording the vehicles "current" configuration, post-build configuration changes responses to manufacturer upgrades, supersession, etc. The advisability and liability issues of permitting any upgrades to the ID are significant concerns, present VIN plates are considered none transferable or changeable.

Another possibility is the use of CD-ROMs which could be supplied with the vehicle in a secure location and contain all the identification and build data. These could be cross checked with the vehicle systems by dealers, at MOT for example, and are not easily changed by unauthorised tampering.

Finally, a network of dealers can be on-line to the vehicle manufacturer, which holds a database for every car it built. A dealer can then access the information instantly via a datalink, but the vehicle manufacturer will have control over the maintenance of the database.

6.7.2 Vehicle Characterisation

Characterising what components are fitted to a vehicle allow the appropriate control strategies to be employed. This breaks down into two main issues. What the control unit has the ability to control and what components are fitted to a vehicle.

What the control unit can control is a function of the build of the unit and can therefore be embedded with the unit's program. Access to the supported features via a communications link is recommended. What components are fitted to a vehicle is a function of the vehicle manufacturing process and at time of vehicle build the programming of the control unit with the components available under its control should be considered. In this way the vehicle environment is known to the control strategies. If a control unit is affected by vehicle fit items not directly controlled by it then this should also be programmed, e.g. whether a vehicle has manual or automatic gearbox may impact on the engine management system.

Vehicle Identification - VIN or electronic identification. Ideally capable of identifying up to the current owner and down to which (Sub)systems are fitted.

Enhancement of anti-theft security is possible by distributing system functionality around the vehicle in different control units. A Vehicle Identification Code could be received, transmitted and compared by every control unit connected to the communications bus, resulting in immobilisation of the vehicle if an incorrect code were detected.

The Environmental Protection Agency in the USA mandates that the following information is needed for emissions certification :

- Year
- Manufacturer (MFR)
- Engine Displacement
- Vehicle Class
- Fuel System
- Number of valves per cylinder
- Cycle/Fuel Type
- Standards
- Catalyst Type
- OBD data

Also in the USA, the SAE's Vehicle Electronic ID Task Force J1724 [15] is defining a more comprehensive list of information that is needed for vehicle identification.

J1724 Vehicle Content Identification - preliminary list:

- Year (Vehicle Release)
- Manufacturer (MFR)
- Vehicle Make
- Model
- Platform Designation
- Engine (Codem Description)
- Number of Cylinders
- Engine Displacement
- Block Configuration
- Number of Valves per cylinder
- Fuel System
- Induction System
- Transmission
- Suspension
- Emission Standard (CA, Fed,etc.)
- Vehicle Weight
- Vehicle Test Weight
- Horsepower
- Steering
- OBD Standards
- Axle Ratio
- Tyre Size
- Drive Type (FWD, RWD, etc.)
- Emission Critical Parts (per EPA)
- PCM Calibration Numbers and Revisions
- Body Style
- Fuel Type
- Brakes

The list could also include software release numbers for each ECU, harness information, checksums, keywords, etc.

This list raises a number of issues :

- Is the vehicle content information "as built" or "as is"?
- Are both required for vehicle diagnosis, service, test and repair.?
- How will the information be updated?
- Could the information be communicated electronically to other vehicle systems?

The above table would need to fully describe the vehicle, e.g. body style would need to describe whether electric seats fitted, alarm system type , instrument pack, etc. All could have interaction with the rest of the vehicle systems and therefore would require diagnostics and potentially a communications facility.

6.7.3 Outstanding Recall Actions

Recall actions are usually triggered by vehicle model and VIN. Often rectification of faults can be actioned at a standard service. In the absence of an integrated dealer network the service details of a vehicle may be fragmented between a number of dealers. The use of a dealer network would alleviate repeated work but it could be implemented other ways. The services and any other work carried out on a vehicle could be stored within a control unit and

any dealer could then have access to this information. The storage within a control unit is usually limited and problems could arise if the unit was changed.

6.8 Production Line ("On-track") Programming

The ability to reprogram or part-program an ECU gives the potential for greatly reducing manufacturing complexity and ultimately costs. On-track programming allows the manufacture of a standard control unit which is subsequently programmed for the intended application. This can be achieved in two ways :

- Programming the unit on the ECU manufacturing line.
- Programming the unit on the vehicle manufacturing line .

The first can possibly reduce ECU manufacturing costs for the component suppliers. The second option can achieve the reduction in inventory at the end of the track, and the programming can be either full downloading of code, or the setting of flags to select different areas of code for different functionality. Obviously integrity is an issue for both.

A third possibility lies with programming units in the field when replacement units are required. This would reduce the vehicle’s cost of ownership due to reduced inventory costs. Problems occur due the mechanism to program an ECU because it would be open to wider abuse.

Vehicle identification is vitally important for production line programming to ensure that the vehicle is configured by the correct program.

Process time must also be considered, as it is not feasible to allow the time for programming to become a bottleneck on the production time. Hence if it does take a significant time to download and check the program it may be necessary to have several "programming stations" in parallel.

Finally, the link between vehicle and programming equipment must be robust so as to prevent corruption of the code during transfer. It is recommended that a communications link which offers low residual error probability is used, that checks are made by reading back the code and that appropriate checksums are used.

7. References

[1] Ward, P. & Mellor, S., *Structured Development for Real-Time Systems,* Yourdon Press, 1986. Volume I - Introduction and Tools. Volume II - Essential Modelling Techniques. Volume III - Implementation Modelling Techniques.

[2] Hately, Derek J. & Pirbhai, Imtiaz, *Strategies for Real-Time System Specification.*

[3] Kiencke, W., *Distributed Realtime Processing in Automotive Networks*, SAE Technical Paper no 900696, 1990.

[4] International Electrotechnical Commission, IEC/SC65A/(Secretariat 123), *Functional safety of electrical/electronic/programmable electronic systems: Generic Aspects, Part 1: General Requirements,* Draft for comment, 1991.

[5] International Electrotechnical Commission, IEC/SC65A/(Secretariat 122), *Software for computers in the application of industrial safety-related systems.* Draft for comment, 1991.

[6] Nakamura S., Isobe T. & Hirabayashi, Y., *The High Speed In-Vehicle Network of Integrated Control System for Vehicle Dynamics,* SAE Technical Paper no 910463, 1991.

[7] McLaughlin, R., *EMC Susceptibility of a CAN Car,* SAE Technical Paper no 932866, 1993.

[8] Dais S. & Chapman M., *Impact of Bit Representation on Transport Capacity and Clock Accuracy in Serial Data Streams,* SAE Technical Paper no 890532, 1989.

[9] Wang Z., Lu H. & Stone M., *A Message Priority Assignment Algorithm for CAN Based Networks,* ACM Paper No 089791-472-4/92/0002/0025.

[10] Unruh, J., Mathony H. & Kaiser K., *Error Detection Analysis of Automotive Communication Protocols,* SAE Technical Paper no 900699, 1990.

[11] Cullyer, W.J., Goodenough, S.J., & Wichmann, B.A., *The choice of computer languages for use in safety-critical systems*, Software Engineering Journal, March 1991.

[12] EC DRIVE I V1050 *DRACO.*

[13] EC DRIVE I V1033 *AUTOPOLIS.*

[14] EPA NPRM (OBD) and CARB 1968.1 (OBD II), *Diagnostic documents.*

[15] Society of Automotive Engineers, SAE J1724, *Vehicle Electronic ID Task Force*, meeting minutes December 8, 1993.